UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE VISUALIZACIÓN DE VÍDEO MULTIVISTA PARA UN DISPOSITIVO DE GAFAS DE REALIDAD VIRTUAL

JAVIER CUBELOS ORDÁS

2016

TRABAJO FIN DE GRADO

TÍTULO:	**Desarrollo de un sistema de visualización de vídeo multivista para un dispositivo de gafas de realidad virtual**
AUTOR:	**D. Javier Cubelos Ordás**
TUTOR:	**D. Pablo Carballeira Lopez**
PONENTE:	**D. Francisco Morán Burgos**
DEPARTAMENTO:	**Departamento de Señales, Sistemas y Radiocomunicaciones**

TRIBUNAL:

Presidente:	**D. FERNANDO JAUREGUIZAR NÚÑEZ**
Vocal:	**D. JULIÁN CABRERA QUESADA**
Secretario:	**D. CARLOS ROBERTO BLANCO ADÁN**
Suplente:	**D. LUIS SALGADO ÁLVAREZ DE SOTOMAYOR**

FECHA DE LECTURA: _____

CALIFICACIÓN: _____

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DESARROLLO DE UN SISTEMA DE VISUALIZACIÓN DE VÍDEO MULTIVISTA PARA UN DISPOSITIVO DE GAFAS DE REALIDAD VIRTUAL

JAVIER CUBELOS ORDÁS

2016

Resumen

El desarrollo de las tecnologías de captura de contenido audiovisual, y la disminución del tamaño de sensores y cámaras, hace posible, a día de hoy, la captura de escenas desde múltiples puntos de vista simultáneamente, generando distintos formatos de vídeo 3D, cuyo elemento común es la inclusión de vídeo multivista.

En cuanto a las tecnologías de presentación de vídeo 3D, actualmente existen diversas opciones tecnológicas, entre las cuales empiezan a tomar una gran importancia las gafas de realidad virtual, también conocidas como *Head-Mounted Devices* (HMD). Este tipo de gafas principalmente han sido utilizadas para la visualización de vídeo panorámico (o 360). Sin embargo, al permitir localizar al usuario (posición de la cabeza y orientación), habilitan también la posibilidad de desarrollar sistemas para la visualización de vídeo multivista, ofreciendo una funcionalidad similar a la de los monitores autoestereoscópicos.

En este Trabajo Fin de Grado se ha desarrollado un prototipo de un sistema que permite visualizar vídeo 3D multicámara en las Oculus Rift, un dispositivo HMD. Este sistema toma como entrada una secuencia de vídeos multivista (real o generada por ordenador) y permite, a partir de la información proporcionada por los sensores de las Oculus Rift, variar el punto de vista adaptándolo a la posición del usuario.

El sistema desarrollado simula la visualización de un monitor autoestereoscópico y es parametrizable. El sistema permite variar una serie de parámetros como la distancia interocular o la densidad de cámaras, y dispone de varios modos de funcionamiento. Esto permitirá que el sistema pueda utilizarse para distintas secuencias *Super MultiView* (SMV), volviéndolo a la vez útil para la realización de pruebas subjetivas de calidad de experiencia.

Palabras clave

Vídeo 3D, multivista, multicámara, Oculus Rift, Head-Mounted Devices, HMD, Super MultiView, SMV, pruebas subjetivas.

Abstract

The development of the capture technologies, and the decrease of the size of the sensors and cameras, makes it possible for us, today, to capture a scene from multiple views simultaneously, creating different 3D video formats, whose common element is the inclusion of multiview video.

Concerning the technologies of 3D presentation, several technological options coexist nowadays, among which the virtual reality glasses, also known as Head-Mounted Devices (HMD), are starting to take a big importance. This kind of glasses have been mainly used for the visualization of panoramic video (or 360º). However, allowing to locate the user (head position and orientation), they allow also the possibility of developing systems for displaying multiview video, offering a similar functionality as the autostereoscopic displays.

The main goal of this project has been the development of a prototype of a system that allows to view 3D multi-camera video in the Oculus Rift, a HMD. This system takes a sequence of multiview videos (real or generated by a computer) as an input, and, thanks to the information provided by the Oculus Rift's sensors, allows to vary the point of view adapting it to the user position.

The system simulates the visualization of an autostereoscopic display and it's highly configurable. The system makes possible the variation of some parameters, as the stereopsis or the density of cameras, and has different working modes. This makes the system useful for different Super MultiView (SMV) sequences and makes it suitable to be used for subjective tests of quality of experience.

Keywords

3D video, multi-camera, multiview, Oculus Rift, Head Mounted-Devices, HMD, Super MultiView, SMV, subjective tests.

Agradecimientos

A mis padres Pepe y Anuska, por haberme hecho ser quien soy, apoyándome y cuidándome desde el primer día. A mis hermanos Pepe y Jorge, por la compañía, la complicidad y el cariño especial. Al resto de mi familia, y en especial a mis abuelos Ángel y Carmelis que nos dejaron hace poco, por cuidarme y hacerme sentir orgulloso de venir de dónde vengo.

A mis amigos de siempre, y en especial a Nacho, por estar siempre ahí, crecer juntos y compartir tan buenos momentos. A mis artistas de teatro, por crearme una vía de escape de la monotonía y por todo su apoyo y frescura. A mis amigos de la universidad, por tantos recuerdos y haberme acompañado en este cambio en mi vida. A Álvaro, por haberme ayudado tanto en los momentos más difíciles y frustrantes de este proyecto.

A mis compañeros del Grupo de Tratamiento de Imágenes, por hacerme sentir como uno más, por haberme hecho desarrollar un interés especial por esta rama y por orientarme y ayudarme durante la realización de este proyecto. Y por supuesto, quería agradecer a Pablo por ser mi hermano mayor del departamento, por creer en mí y por ayudarme tanto, siempre con una paciencia especial.

Muchas gracias a todos.

Índice

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Motivación

El desarrollo de nuevos formatos de vídeo ha tomado gran importancia en el sector audiovisual. Tanto los formatos de alta resolución (2K, 4K…) [1], como los formatos tridimensionales (vídeo estereoscópico o multivista) [2] están orientados a ofrecer a los usuarios una mayor sensación de inmersión a la hora de visualizar contenido audiovisual.

Este avance no hubiese sido tan notable de no verse acompañado por el correspondiente desarrollo de las tecnologías de captura, especialmente en la disminución del tamaño de los sensores que conlleva que se puedan desarrollar arrays multicámara [3]. También es notable en el mundo de los displays, con la reciente llegada al mercado de los televisores autoestereoscópicos [4] (ej. Phillips, Dimenco), los Super MultiView displays [5] (SMV, ej. Holografika) y de los cascos de realidad virtual [6] (o Head-Mounted Devices, HMD), principalmente orientados a la visualización de contenido 360º estéreo [6] o juegos inmersivos.

Existen por tanto, una gran variedad de formatos tridimensionales y displays para visualizar el contenido. Típicamente, cada contenido está adaptado al display en el que se visualizará, como ocurre con el contenido multivista y los televisores autoestereoscópicos, o el contenido 360º y los HMDs. En este proyecto se abordará un campo poco explorado hasta el momento, el desarrollo de herramientas para la visualización de contenido multivista en HMDs.

1.2. Objetivos y contribuciones

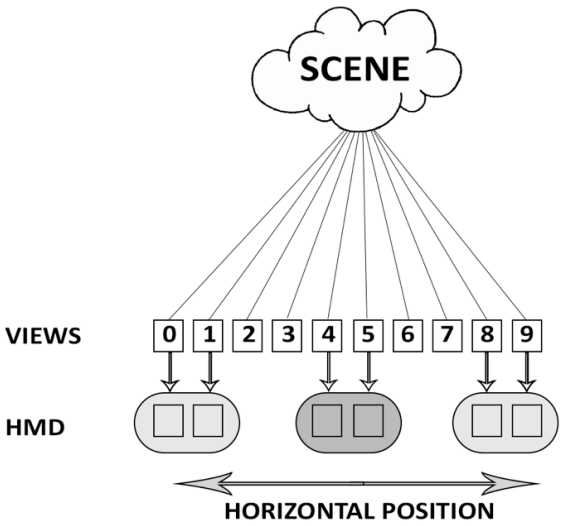
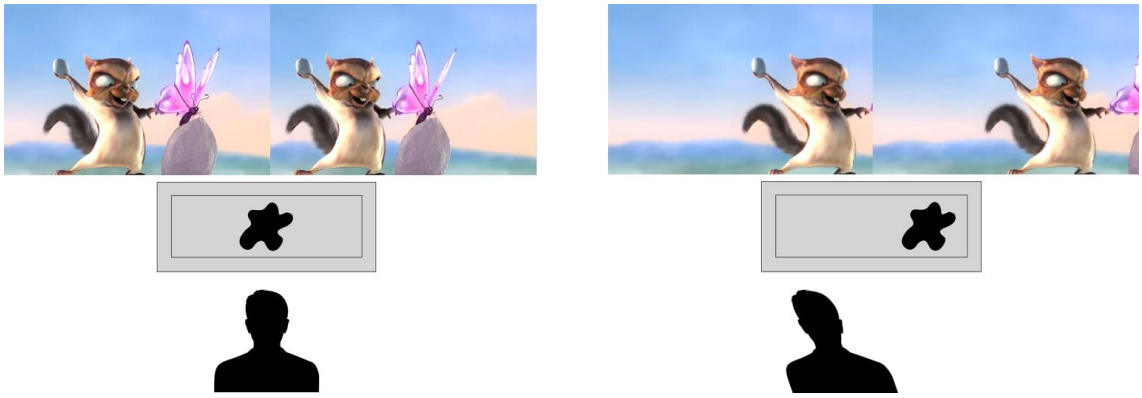


Figura 1 - Diagrama explicativo básico del sistema desarrollado

El objetivo de este proyecto ha sido el desarrollo de un prototipo de un sistema de visualización de vídeo 3D multivista para un dispositivo HMD como Oculus Rift. El sistema desarrollado simula la visualización de un monitor autoestereoscópico, adaptándose a la vez a las características peculiares de un HMD. Dicho sistema permite variar el punto de vista en función del posicionamiento del usuario.

En la Figura 1 se observa un diagrama descriptivo del funcionamiento básico del sistema desarrollado. A grandes rasgos, las Oculus Rift obtienen información de la posición de la cabeza del usuario dentro del array de cámaras, y, en función de dicha posición, presentan la vista procedente de una u otra cámara en el display. Al tratarse de visión estéreo, estaremos trabajando en paralelo con dos vídeos de dos cámaras que representan las dos vistas de las que dispone el usuario (ojo izquierdo y ojo derecho) y cada una de ellas en la parte correspondiente del display de las Oculus Rift. En la Figura 2 se presenta un ejemplo de las vistas que visualizaría el usuario para una posición central y para una posición máxima izquierda del array de cámaras.



a) b)

Figura 2 - a) Ejemplo de par de vistas visualizado para la posición central del array de cámaras
b) Ejemplo de par de vistas visualizado para la posición límite izquierda del array de cámaras

Una de las funcionalidades principales que presenta sistema desarrollado es la posibilidad de utilizarlo para la realización de pruebas subjetivas de calidad de experiencia [8]. Por eso, otro de los objetivos ha sido volver el sistema lo más parametrizable posible, con el fin de evaluar la influencia de diferentes configuraciones en la calidad de experiencia y de poder utilizar el sistema para distintas secuencias multivista. Así mismo, se ha llevado a cabo, una primera evaluación subjetiva de los distintos parámetros configurables del sistema

El sistema desarrollado es multiplataforma, siendo compatible con cualquier sistema operativo soportado por el HMD utilizado, y se ejecuta un navegador. Estas dos características del sistema facilitan su distribución; pudiendo ser alojado incluso en un servidor web al que los usuarios pudieran acceder remotamente.

Este proyecto ha dado pie a una contribución al grupo ISO/IEC SC29WG11 Moving Picture Experts Group (MPEG) presentada durante el *115th meeting of MPEG* en Ginebra (Suiza) en el mes de Mayo de 2016 [9]. Concretamente en el Ad-Hoc Group FTV, que se encuentra actualmente trabajando en el desarrollo de estándares de compresión para vídeo SMV.

1.3. Estructura de la memoria

En la Sección 2 se introducirán los principales conceptos necesarios para la correcta comprensión del proyecto. Empezaremos con la Sección 2.1 centrándonos en los sistemas audiovisuales tridimensionales, sus principales dispositivos de captura, formatos de representación y, en especial, el formato de vídeo multivista. Así mismo, en la Sección 2.2 se introducirán los distintos displays disponibles para la visualización de este tipo de contenido,

presentando especialmente los cascos de realidad virtual, presentando los distintos tipos de HMD, y estudiando en particular las características del *Development Kit 2* de Oculus Rift [10] (utilizado en la realización del proyecto). A continuación, en la Sección 2.3, se hará una breve descripción de las principales aplicaciones *Free Viewpoint Video* [11] (FVV) disponibles hasta el momento para HMD. Finalmente, en la Sección 2.4 se introducirá Unity [12], la herramienta utilizada durante el desarrollo del sistema.

En la Sección 3 se presentará el sistema desarrollado. En esta sección se empezará con una introducción general del sistema desarrollado (Sección 3.1) y se continuará estudiando más en detalle el funcionamiento de cada uno de los bloques del sistema (Sección 3.2), tanto los bloques que forman parte del bucle de operación del sistema, como los que forman parte de la preparación de la escena 3D o la parametrización de dicho sistema.

En la Sección 4 se abarcará el estudio en detalle de los distintos parámetros del sistema. En esta sección se pondrá a prueba el sistema, analizando los distintos parámetros configurables mediante una serie de pruebas experimentales, que permitirán localizar los límites de cada uno de ellos, y sus valores más adecuados para distintas configuraciones y/o secuencias.

Finalmente, y tras una conclusión descriptiva del objetivo alcanzado y de las posibles líneas futuras de investigación relacionadas con este proyecto (Sección 5), y tras la bibliografía (Sección 6), se presentarán como anexo en la Sección 7: los requisitos del sistema (navegador y formato de vídeos necesarios), y, para terminar, un manual de usuario completo en inglés que explique los principales aspectos del sistema, facilitando su configuración y utilización.

2. TECNOLOGÍAS Y HERRAMIENTAS RELACIONADAS

En esta sección se va a realizar una introducción a los conceptos necesarios para la comprensión del resto del documento. Se empezarán describiendo los sistemas audiovisuales tridimensionales, así como los dispositivos de captura, los formatos de representación característicos de este tipo de contenido y los displays utilizados [13]. A continuación nos centraremos en la descripción de los HMDs y de Unity, entorno de desarrollo utilizado durante todo el proyecto.

2.1. Sistemas Audiovisuales Tridimensionales

2.1.1. Introducción

En los últimos años se ha observado un importante aumento en el interés por la inclusión de la tercera dimensión en el vídeo 2D convencional. Las nuevas tecnologías de adquisición y representación de contenido 3D [14], junto a la convergencia de las tecnologías de gráficos 3D y visión por ordenador, han hecho posible que el vídeo 3D sea, hoy en día, una realidad.

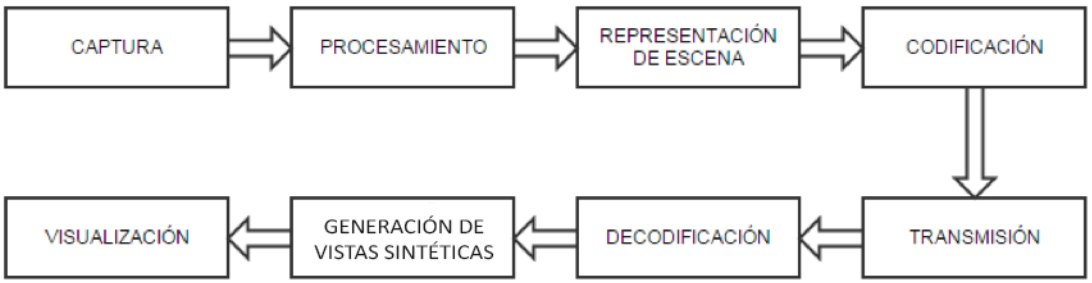


Figura 3 - Diagrama de bloques básico de un sistema audiovisual tridimensional

En la Figura 3 se observa el diagrama de bloques descriptivo de este tipo de sistema, donde se distinguen las siguientes fases: captura, procesamiento, representación de la escena, codificación, transmisión, decodificación, renderizado interactivo y visualización (*display*) [15].

Aunque estos sistemas suelen presentar diferencias en los algoritmos de procesamiento, existe un elemento común entre ellos, y es el uso de múltiples vistas de una misma escena. Estas señales múltiples se transforman en un único formato específico de representación de escena, que permite la síntesis de vistas intermedias virtuales (como veremos en la Sección 2.1.4).

Tras una breve explicación de los tipos de sistemas audiovisuales tridimensionales, nos centraremos en el estudio de los distintos dispositivos y formatos de captura y, de los distintos dispositivos disponibles para la visualización final (*displays*), al ser estos conceptos clave para la correcta comprensión del proyecto.

2.1.2. Free Viewpoint TV y 3D Video

Existen dos grandes categorías de sistemas audiovisuales tridimensionales (Figura 8), que son el *FVV* y el *3D Vídeo (3DV)*. Estos sistemas forman parte de las líneas de investigación en constante desarrollo, al presentar estos mejoras importantes en la experiencia del usuario respecto a los contenidos audiovisuales tradicionales.

Los sistemas FVV permiten que, en una escena real capturada por cámaras reales, el usuario pueda elegir el punto de vista y la dirección de dicha visión. En otras palabras, permite la selección interactiva de la vista visualizada en cada momento, entre una serie de vistas disponibles. Adicionalmente, al haber sido la escena capturada por un número limitado de cámara, el punto de vista seleccionado no tiene por qué ser una cámara real, sino que puede ser una vista sintética creada a partir de las capturadas por cámaras reales.

Los sistemas 3DV, por su parte, tienen como objetivo proporcionar al usuario una sensación de profundidad 3D en la escena observada. Para ello, es necesario que los dos ojos del usuario observen simultáneamente un par estéreo de imágenes. En un par estéreo de imágenes, una vista se encontrará ligeramente desplazada hacia la izquierda y la otra hacia la derecha, siempre tomando como punto de vista central el situado en medio de ambas vistas.

No obstante, no existe una clara separación entre FVV y 3DV y, la distinción entre estos dos sistemas, se justifica por razones históricas y por las diferencias en las áreas de investigación. Esto se debe a que la generación de una señal de vídeo estéreo que proporcione sensación de profundidad 3D también puede ser lograda con las técnicas de síntesis de vistas de los sistemas FVV.

De hecho, los sistemas más avanzados tienden a combinar las funcionalidades de ambos tipos de sistemas, permitiendo combinar la sensación de profundidad con la libre selección de diferentes puntos de vista, como ocurre en la extensión de los sistemas 3DV a configuraciones multivista donde cada punto de vista proyecta un par de vistas estéreo, o como ocurre con los HMD como las Oculus Rift (Sección 2.2).

2.1.3. Dispositivos de captura

Hoy en día, existen una gran variedad de cámaras y sensores que permiten capturar nuevas formas de datos visuales junto a las dimensiones del espacio. Destacan entre esta gran variedad dos tipos de sistemas de captura: los sistemas de captura divergentes y los sistemas de captura convergentes.

Figura 5 - Ejemplo de sistema de captura divergente

Figura 4 - Ejemplo de sistema de captura convergente

Como vemos en la Figura 5, los sistemas de captura divergentes presentan las cámaras orientadas en direcciones divergentes respecto a la posición central, existiendo una baja redundancia entre cámaras. En cambio, los sistemas de captura convergentes (Figura 4) presentan las cámaras orientadas a un mismo punto del espacio,

incluyen los sistemas con cámaras paralelas, y presentan una alta redundancia entre cámaras. Se distinguen varios tipos en cada uno de los sistemas de captura:

- **Sistemas de captura divergentes:**
  - **Cámaras para contenido 360VR:** Cámaras especialmente configuradas para la captura de contenido 360VR, capaces de grabar contenido esférico y estereoscópico [16].
  - **Cámaras con 6 grados de libertad:** Cámaras capaces de capturar contenido en los 6 grados de libertad, las 3 rotaciones características de los sistemas de captura 360º, y adicionalmente, 3 posiciones que permiten el movimiento por la escena capturada [17].

- **Sistemas de captura convergentes:**
  El vídeo multivista es capturado por un array de cámaras, las cuales pueden estar configuradas espacialmente de diferentes formas, dependiendo de las características del sistema en cuestión. Algunos de los sistemas multicámara más utilizados son los siguientes:
  - **Array lineal y array planar:** arrays en los que las cámaras están dispuestas en paralelo, formando una "línea" o "plano" de cámaras, respectivamente. Este tipo de arrays limita la selección de puntos de vista a un solo plano, pero la configuración de las cámaras permite la estimación de la profundidad de la escena (Figura 6).

Figura 6 - Ejemplo de array lineal y array planar (Universidad de Nagoya)

  - **Array en arco:** sistemas multicámara dispuestos en torno a la escena en forma de arco. Este tipo de arrays permiten navegar entre vistas alrededor de la escena, sin restringirse a un único plano como pasa con los arrays lineales o planares (Figura 7).

Figura 7 - Ejemplo de array en arco (Universidad de Nagoya)

2.1.4. Formatos de representación

Para conseguir las funcionalidades de 3DV es necesario un formato de representación de datos que permita la generación de vistas correspondientes a puntos de vistas virtuales. Hoy en día existe un gran abanico de formatos de datos que habilitan estas funcionalidades [18]; yendo esta diversidad desde formatos relacionados con los gráficos 3D hasta formatos de datos puramente basados en imágenes como el vídeo multivista (Figura 8), en el cual se dispone de múltiples vistas de una misma escena.

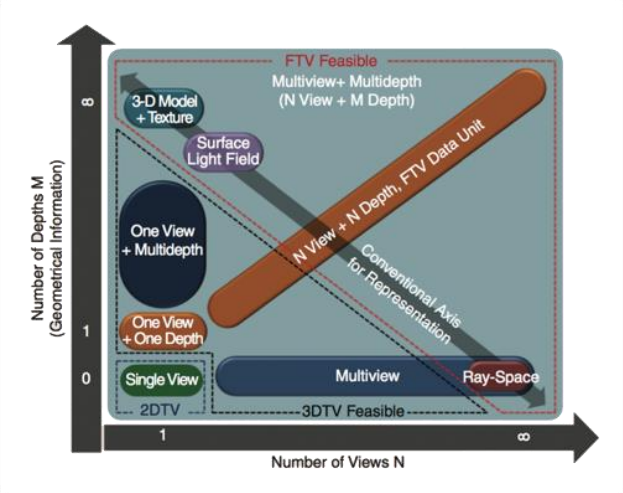


Figura 8 - Formatos de vídeo en función del número de vistas e información de profundidad [19]

Precisamente, uno de los formatos más utilizados es el que une el vídeo multivista con información de la profundidad de objetos de la escena respecto a una cámara (o varias). Este tipo de formatos son los formatos conocidos como vista más profundidad (*view+depth*) [20] y permiten la generación de vistas adicionales en posiciones virtuales, posiciones en las que realmente no existía una cámara durante la captura [21].

Si no se dispone de información extra sobre la profundidad, resulta complicado el crear nuevas vistas virtuales, por lo que, en estos casos, será importante que el array tenga las cámaras necesarias para abarcar la escena deseada manteniendo una densidad de cámaras que no deteriore la suavidad en la transición entre vistas. Esta relación de dependencia, junto a otros factores como el avance de los sistemas de captura (Sección 2.1.3) y la eficiencia de compresión característica del estándar HEVC [22], ha dado lugar a la llegada al formato SMV (*Super Multiview Video*) [23], el cual contiene un rango de vistas que en las configuraciones actuales incluye más de 80 vistas (como veremos en la Sección 4.1.2).

2.2. Displays

2.2.1. Tipos

Hoy en día existe una gran diversidad de dispositivos para la visualización de contenido audiovisual tridimensional. A continuación, describiremos los principales tipos de dispositivo que podremos encontrar para este tipo de sistemas:

- **Sistemas estereoscópicos**: [24] este tipo de displays obliga al usuario a utilizar un dispositivo para la visualización de contenido, como unas gafas que aseguren que las vistas izquierda y derecha sean visualizadas por el ojo adecuado. Estos sistemas están implantados en el mercado y, por ejemplo, dominan el sector del cine 3D.

No obstante, los sistemas estereoscópicos presentan siempre la misma perspectiva de la escena 3D al observador, independientemente de su posición.

- **Sistemas autoestereoscópicos**: estos displays hacen innecesaria la utilización de dispositivos intermedios como las gafas para la correcta visualización del 3D; permitiendo mandar la imagen correcta a cada ojo del observador. Existen dos grandes tipos de sistemas autoestereoscópicos:
    - **Sistemas autoestereoscópicos tradicionales**: estos sistemas trabajan sólo con dos vistas, enviando una imagen distinta a cada ojo. Esto se realiza mediante una barrera de paralaje (o lentes) que se sitúa frente al monitor e interrumpe el haz de luz selectivamente permitiendo que cada ojo reciba únicamente la imagen que le corresponde. Este enfoque permite percibir la sensación de profundidad. No obstante, estos sistemas funcionan exclusivamente con un punto de vista que será fijo independientemente de la posición del usuario.
    - **Sistemas autoestereoscópicos multivista**: estos sistemas nacieron para resolver el problema anterior, permitiendo variar el punto de vista si el usuario cambia de posición. Esto se logra puesto que los sistemas autoestereoscópicos multivista trabajan con varias imágenes visibles en diferentes ángulos, permitiendo que en diferentes posiciones el usuario visualice diferentes pares de vistas.

- ***Head-Mounted Devices*:** también llamados cascos/gafas de realidad virtual, son los dispositivos inmersivos con posicionamiento de cabeza más conocidos del mercado. Estos dispositivos poseen uno o dos pequeños displays ópticos, dependiendo de si se trata de un HMD monocular o binocular. Permite reproducir imágenes, tanto sintéticas como reales, ocupando parcialmente (realidad aumentada) o completamente (realidad virtual) el campo de visión del usuario.

  Esta característica facilita la distinción entre los dos principales tipos de HMD:

  - **Gafas/cascos de realidad aumentada (o HMD ópticos):** estos dispositivos permiten al usuario visualizar su entorno y a su vez introducen objetos sintéticos, creando una combinación entre realidad y virtualidad conocida como realidad aumentada (Figura 9).

Figura 9 - Google Glass:
Ejemplo de casco de realidad aumentada

  - **Gafas/cascos de realidad virtual:** estos dispositivos ocupan la totalidad del campo de visión del usuario, permitiéndole abstraerse de su

Figura 10 - Oculus Rift DK2:
Ejemplo de casco de realidad virtual

entorno y produciéndole una sensación de inmersión completa en el mundo virtual (Figura 10).

2.2.2. El caso de las Oculus Rift

Centramos el análisis de las características de las Oculus Rift en el modelo utilizado a lo largo de todo el proyecto, el *Development Kit 2* (DK2). Este modelo fue lanzado en marzo de 2014 y ha sido el segundo y último kit de desarrolladores que ha presentado Oculus. Finalmente, en febrero de 2016 lanzó la versión oficial de las Oculus Rift para consumidores (CV1), en la que no entraremos a detallar al no ser las utilizadas durante la realización de este proyecto.

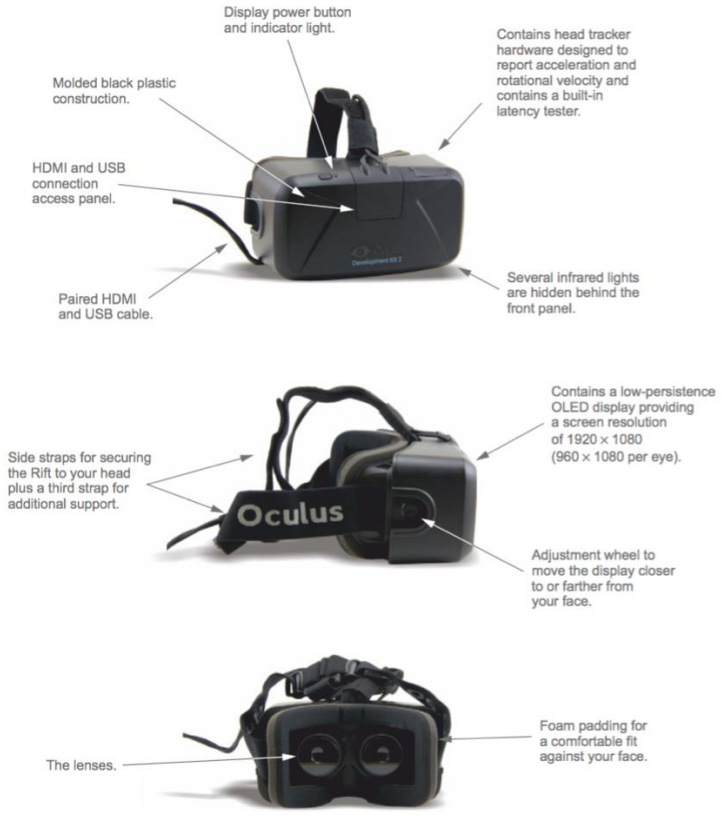


Figura 11 - Componentes de las Oculus Rift DK2 [10]

El DK2 (Figura 11) está compuesto principalmente por un display OLED de persistencia baja con una resolución total de 1920x1080 píxeles, lo que equivale a una resolución de 960x1080 píxeles por ojo (Figura 12), y con una frecuencia de refresco de 75 Hz. El aumento de la resolución respecto al modelo anterior (DK1) conlleva que las imágenes se vean más claras y nítidas. La baja persistencia y alta frecuencia de refresco contribuyen eliminando gran parte del desenfoque de movimiento (*motion blur*), desenfoque presente típicamente en este tipo de dispositivos cuando se varía el punto de vista.

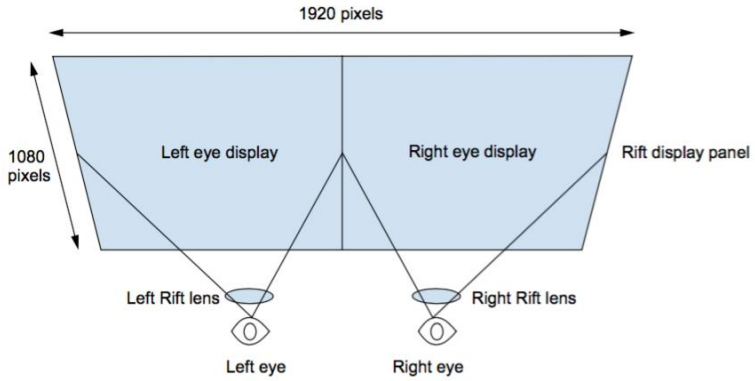


Figura 12 - Display de las Oculus Rift DK2 [10]

Las Oculus Rift contienen una serie de sensores que incluyen un giroscopio, acelerómetro y magnetómetro. Adicionalmente, desde la versión DK2, incluyen también una cámara externa (Figura 13) que será la responsable de realizar un seguimiento de la posición de la cabeza. El giróscopo presente en las DK2 es el encargado de determinar la orientación, mientras que la cámara posicional externa se encarga de determinar la posición. Combinando la información de estos sensores se puede determinar el movimiento de la cabeza del usuario en el mundo real y sincronizar la vista virtual presentada al usuario en tiempo real.

Figura 13 - Cámara posicional de las Oculus Rift DK2 [10]

Cabe destacar el requisito de las gafas de recibir una imagen estéreo en el display dividido, con el fin de enviar una imagen a cada uno de los ojos del usuario. Así mismo, es necesaria aplicar una serie de transformaciones y una corrección de distorsión para cada ojo con el fin de cancelar la distorsión característica de las lentes del dispositivo [25].

2.3. Aplicaciones FVV para HMDs

Actualmente, las aplicaciones para HMDs se centran en el mundo de los juegos inmersivos [26] y en la visualización de contenido audiovisual.

No obstante, a menudo estas aplicaciones exigen unas características técnicas que no todos los HMDs poseen. Esto suele pasar con los HMDs inalámbricos como las Samsung Gear VR [27], los cuales únicamente disponen de giróscopo y utilizan un teléfono móvil como computadora y

display. Este tipo de HMD presenta ciertas limitaciones y está siendo principalmente explotado para la visualización de contenido 360º (divergente), donde el usuario es capaz de cambiar de punto de vista pero no desplazarse por la escena.

El desarrollo de sistemas para la visualización de contenido convergente en HMDs es todavía limitado. Por ejemplo, el grupo MPEG ha empezado recientemente a investigar y desarrollar la visualización de contenido multivista convergente [28], habiéndose realizado ya los primeros experimentos de visualización de secuencias SMV con unas Oculus Rift [29]. No obstante, estos experimentos no han concluido hasta el momento en una aplicación que permita visualizar todo tipo de secuencias SMV, y este es el objetivo perseguido por el proyecto aquí descrito.

2.4. Unity

2.4.1. Introducción a Unity

En esta sección presentamos Unity, el entorno escogido para el desarrollo del sistema. Unity ha sido elegida como la plataforma para el desarrollo del sistema, principalmente debido a su perfecta integración con distintos tipos de HMDs, y en especial con las Oculus Rift.

Unity es una plataforma de desarrollo de videojuegos multiplataforma líder en el sector. Su perfecta integración con los distintos sistemas operativos (Windows, OS X y Linux), la facilidad de migración de un sistema a otro y el gran abanico de plataformas de salida que tiene (Windows, OS X, Linux, Xbox, Playstation, iOS, Oculus, WebGL…) le han hecho adquirir la reputación que hoy en día posee.

El principio de funcionamiento de Unity es simple; al igual que gran parte de las plataformas de desarrollo de videojuegos utilizadas hoy en día, basa la implementación en un escenario 3D totalmente configurable. El usuario puede crear sus propios objetos 3D o importarlos de otros programas como los ya mencionados. También puede situar la cámara/jugador en la escena, iluminarla... Una vez la escena 3D ha sido creada (Figura 14), Unity funciona gracias a los *scripts* (en C# o javascript) que el usuario puede añadir a los objetos.

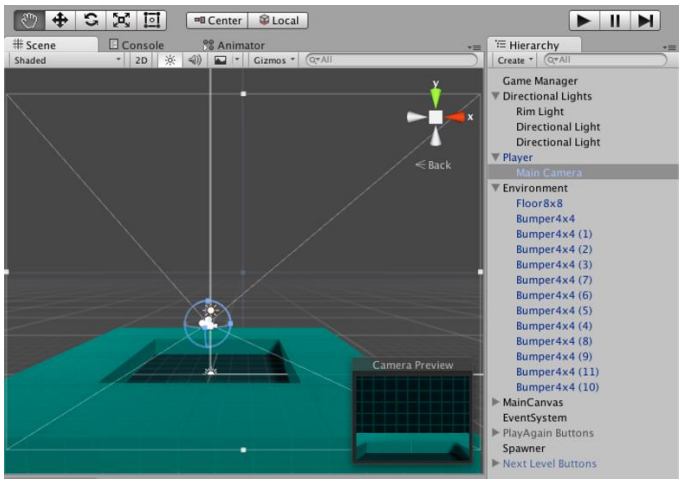


Figura 14 - Ejemplo de escena en Unity

2.4.2. Integración con Oculus Rift

Oculus facilita a los desarrolladores un paquete de integración con Unity 5 [30]. Este paquete de integración incluye una serie de scripts, materiales, texturas y objetos 3D prefabricados que hacen posible el desarrollo para este tipo de dispositivos.

Dentro de este paquete de integración destacan principalmente los objetos prefabricados, que incluyen los script necesarios para simular el comportamiento de un HMDs y hacer posible la comunicación entre, en este caso, las Oculus Rift y Unity. Los dos objetos prefabricados son:

- *OVRCameraRig:* es un objeto que sustituye la cámara normal de Unity y que incluye todos los scripts necesarios para poder visualizar la escena con las Oculus Rift (inicialización del dispositivo, posicionamiento y orientación del usuario, deformaciones necesarias…). Además, está compuesta por tres sub-cámaras (denominadas *anchor* en Unity) que representan las vistas que se observan en el ojo izquierdo y derecho de las Oculus en todo momento, y una cámara virtual adicional localizada en el centro de las dos anteriores.

- *OVRPlayerController:* es el elemento más utilizado para moverse en un entorno virtual. Se trata de un personaje 3D el cual tiene adjuntada una *OVRCameraRig* de las ya mencionadas. Incluye una cápsula física, un sistema de movimiento y los scripts necesarios que permiten que el personaje se mueva por la escena 3D.

Aparte de las características especiales de estos objetos mencionadas anteriormente, la cámara prefabricada de Oculus funciona como un objeto más de Unity. Eso nos permitirá poder utilizar funciones de Unity para poder obtener la posición y orientación de las Oculus Rift en todo momento, e integrar la cámara de Unity con el tracker y giróscopo de las gafas.

Para la realización del proyecto utilizaremos únicamente la función para obtener la posición de las Oculus y en concreto la coordenada *x* que permitirá localizar los movimientos de cabeza horizontales del usuario, que serán la clave del sistema.

Además de este paquete de integración, y como veremos en la Sección 3.2.2.3, ha sido necesario utilizar un paquete adicional para poder mantener una correcta sincronización entre las vistas. Aunque las consecuencias que conlleva la utilización de este paquete se vayan a explicar en detalle en el apéndice 7.1, si es importante mencionar, llegados a este punto, la necesidad que conlleva de tener que crear un proyecto ejecutable en navegador, que podrá ser por tanto alojado en un servidor al que se acceda remotamente. Esto facilitará considerablemente su distribución, volviendo también el sistema multiplataforma, compatible con cualquier sistema operativo que soporte el navegador necesario y que dé soporte a Oculus Rift (ver apéndice 7.1).

3. DESARROLLO DEL SISTEMA

En esta sección vamos a describir el funcionamiento del sistema desarrollado. Para ello, y partiendo de una introducción general del sistema, pasaremos a analizarlo mediante una estructura de bloques, en la cual nos detendremos para ir explicando los principales conceptos de cada uno de los bloques implicados.

3.1. Introducción al sistema

El sistema implementado permite la inmersión del usuario en una escena 3D que simula un display autoestereoscópico que varía el punto de vista presentado con la posición de la cabeza, volviéndolo adecuado para la visualización de vídeo multivista.

Figura 15 - Escena 3D utilizada en el sistema

Como vemos en la Figura 15, para simular el display autoestereoscópico se crean dos pantallas situadas en la misma posición en la escena 3D en las que se texturizarán las vistas correspondientes a la posición de la cabeza del usuario. Estas vistas serán finalmente las vistas visualizadas en cada uno de los dos ojos, al ser cada una de ellas visible únicamente por uno de los ojos.

Figura 16 - a) Par de vistas estéreo para un usuario situado en la posición central
b) Par de vistas estéreo para un usuario situado en el límite izquierdo del array (40cm a la izquierda)

En la Figura 16 se observa un ejemplo de cómo reacciona el sistema al movimiento del usuario, mostrando el par de vistas estéreo *a)* para un usuario situado en la posición central, y el par de vistas *b)* correspondiente a la posición límite izquierda del array (desplazamiento de 40cm hacia la izquierda en este caso).

Un requisito clave del sistema es la fluidez, por lo que es importante que se minimice la latencia entre el movimiento de cabeza del usuario y el correspondiente cambio de vista. Así mismo, para que el usuario pueda vivir una experiencia 3D confortable, también es vital mantener una correcta sincronización en todo momento entre las vistas de los dos ojos.

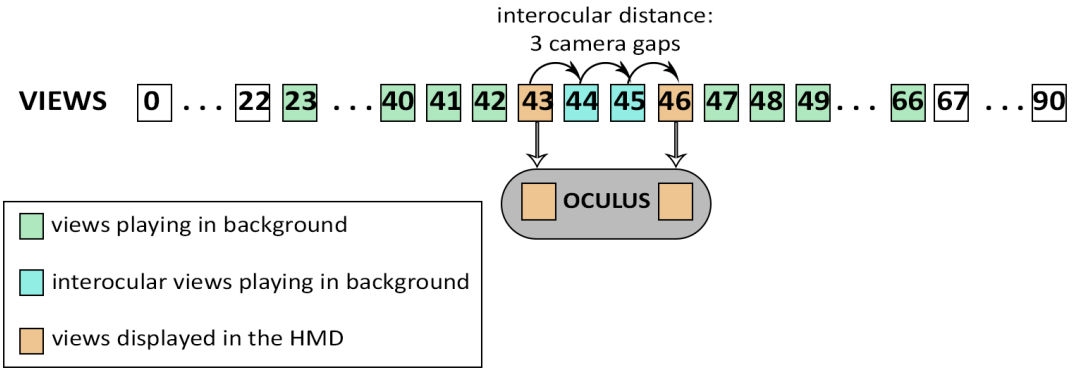


Figura 17 - Ejemplo de vistas en reproducción en segundo plano para un usuario situado en la posición central

Para cumplir estos requisitos, el sistema está, además de las vistas texturizada en las pantallas, constantemente reproduciendo en segundo plano (sin texturizar en las pantallas), y sincronizando, un conjunto de vistas alrededor de la posición actual del usuario (Figura 17). Así, si el usuario se mueve a una vista del conjunto, el nuevo par de vistas estará preparado para ser texturizado en las pantallas sin retardos perceptibles.

El posicionamiento de usuario se realiza en cada actualización de cuadro e indica el inicio del bucle del sistema. Durante este bucle, y tras la extracción de la posición de la cabeza, se calculan los índices de los vídeos correspondientes a las vistas de la posición extraída, los cuales serán texturizados en las pantallas de la escena mencionadas anteriormente y se actualiza el conjunto de vistas en reproducción en segundo plano.

Así mismo, el conjunto de vistas en reproducción es adaptativo a la posición del usuario: se preparan el mismo número de vistas en cada dirección cuando el usuario se sitúa en posiciones cercanas al centro del array, y el ratio varía cuando el usuario se encuentra cerca de uno de los límites del array, momento en el que pasa a reproducir más vistas en el sentido opuesto al límite.

Además, para aumentar el interés de este sistema para diferentes situaciones experimentales, se ha decidido volverlo lo más parametrizable posible, permitiendo variar una serie de parámetros (distancia interocular, número de vistas en reproducción en segundo plano, distancia de conmutación entre vistas…) que permitan ajustar el sistema a distintas secuencias de vídeo de entrada y a distintas necesidades.

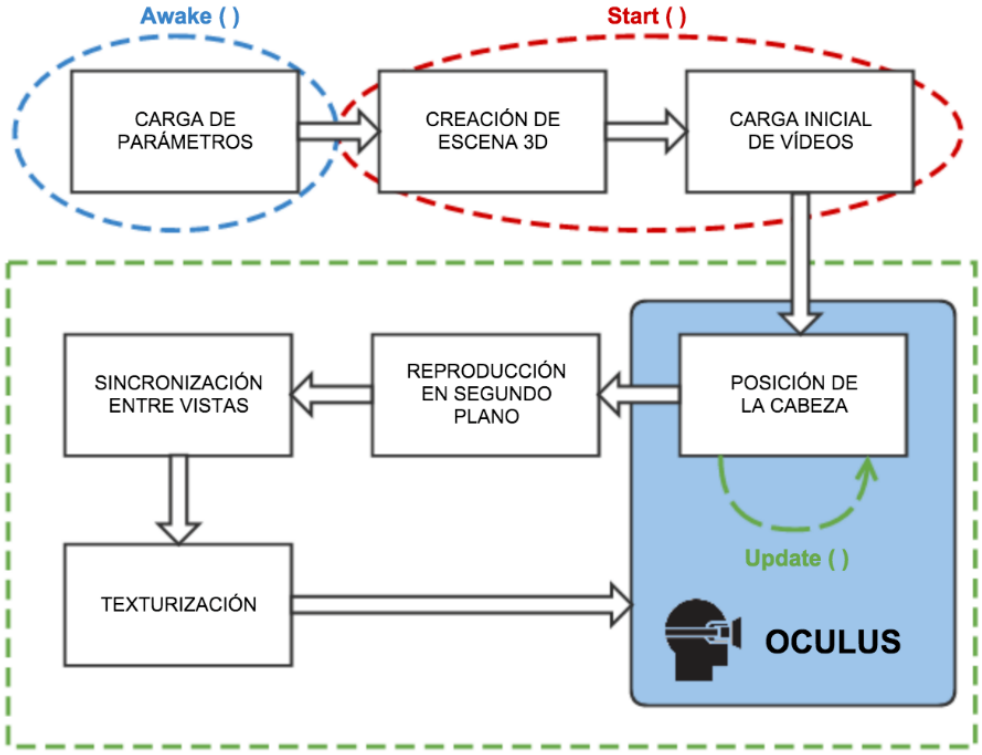


Figura 18 - Diagrama de bloques del sistema

El diagrama de bloques aquí mostrado (Figura 18) representa, a grandes rasgos, las distintas partes implicadas del sistema, que ya han sido mencionadas, y que pasaremos a describir en la Sección 3.2. En la Sección 3.2.1 se abarcará la preparación de la escena 3D, y posteriormente, en la Sección 3.2.2 se describirán los distintos bloques implicados en el bucle de operación del sistema (posicionamiento del usuario, reproducción en segundo plano, sincronización y texturización). Finalmente, en la sSección 3.2.3 se explicarán los distintos parámetros configurables del sistema.

3.2. Descripción detallada del sistema

3.2.1. Preparación de la escena 3D

La escena 3D (que ya hemos visto en la Figura 15) se creará al inicio de la ejecución del sistema, antes del bucle de operación, y estará compuesta principalmente por 5 elementos:

1. La cámara que representa las Oculus Rift, compuesta a su vez de tres sub-cámaras:
   - *CameraL:* cámara correspondiente a lo visualizado en la parte del display correspondiente a la vista del ojo izquierdo
   - *CameraR:* cámara correspondiente a lo visualizado en la parte del display correspondiente a la vista del ojo derecho
   - *CameraMain:* corresponde a la media entre las dos cámaras mencionadas anteriormente.

2. Las dos pantallas superpuestas (situadas en la misma posición) en las que posteriormente se texturizarán los vídeos para cada una de las vistas (*ojo_izquierdo* y *ojo_derecho*).

3. La luz necesaria para iluminar dichas pantallas y eventualmente, el resto de la escena. Se ha escogido una luz direccional que simula la luz solar y es capaz de iluminar uniformemente la totalidad de la escena.

4. Un encapsulador, (Figura 19) aquí llamado *Main Camera*, que no será más que un objeto vacío que creará la relación necesaria entre los objetos mencionados en el código y los incluidos en la escena 3D.

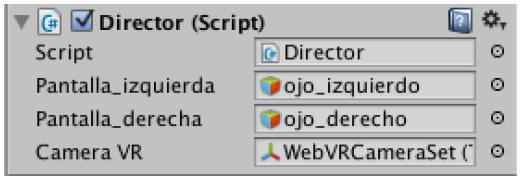


Figura 19 - Relación entre variables y objetos en el encapsulador de la escena

Cabe destacar lo importante que es asociar a cada una de las cámaras responsables de las vistas izquierda y derecha, su pantalla correspondiente; puesto que en ambas pantallas se texturizarán diferentes vistas. Eso se consigue gracias a la opción de crear capas en la escena 3D de Unity. La pantalla correspondiente a cada ojo es puesta en una capa independiente para posteriormente asociar una cámara a cada una de las capas (pantallas). Esto es posible con el parámetro de las cámaras que se llama *Culling Mask* y que permite que nuestro sistema funcione como un display autoestereoscópico, teniendo inicialmente un par de pantallas idénticas situadas en la misma posición.

Así mismo, este sistema se utilizará para secuencias de vídeo con diferentes resoluciones y/o relaciones de aspecto, por lo que es importante escalar estas pantallas para que las vistas texturizadas en ellas mantengan la relación de aspecto original.

Las pantallas se sitúan inicialmente a una distancia de 3H (*Height*), 3 veces la altura del vídeo, de la cámara que representa las Oculus Rift en la escena 3D. Esta distancia es modificable, permitiendo al usuario acercar o alejar dichas pantallas mediante el teclado, con el fin de poder adaptar esta distancia a las diferentes secuencias:

- Tecla 'ARRIBA': acercar pantallas
- Tecla 'ABAJO': alejar pantallas

3.2.2. Bucle de operación

En este sistema existe una tasa de refresco de Unity, que no tiene por qué coincidir con la tasa de cuadros (*framerate)* del vídeo texturizado en cada momento, y que marca el ritmo de ejecución del bucle de operación.

El bucle de operación del sistema es el responsable de la comunicación con el HMD y de la correspondiente texturización y sincronización de vistas. En un primer momento se extrae la posición de la cabeza a partir de la información proporcionada por el HMD. Una vez se conoce el índice de la vista correspondiente a la posición extraída, se actualiza el conjunto de vistas en reproducción en segundo plano y se mantienen sincronizadas. Finalmente, se texturizan en las pantallas de la escena 3D las vistas correspondientes a la posición extraída.

3.2.2.1. Posición de la cabeza

Es el primer bloque del bucle de operación que se ejecuta siguiendo la tasa de refresco de Unity, y tiene como principal objetivo la obtención de la coordenada horizontal correspondiente a la posición actual de las Oculus Rift y el cálculo del índice de la vista correspondiente a la nueva posición.

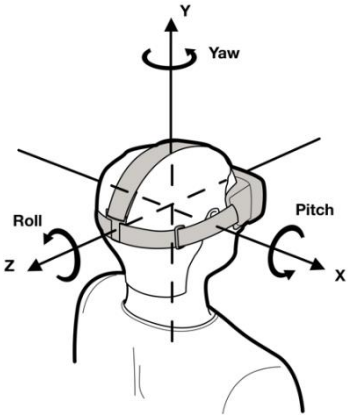


Figura 20 - Sistema de coordenadas de las Oculus Rift

Para la extracción de la posición horizontal se utiliza el sistema de coordenadas de las Oculus Rift (Figura 20), siempre respecto al plano de la cámara de tracking.

La coordenada *x* representa el movimiento horizontal de la cabeza (izquierda o derecha), mientras que la coordenada *y* representa el movimiento vertical de esta (arriba o abajo) y la coordenada *z* el movimiento hacia delante o detrás.

La posición del HMD se devuelve finalmente en forma de vector 3D (x, y, z).

Una vez extraída la posición, nos quedamos con la coordenada *x* por ser la única relevante para nuestro sistema, al estar este centrado en la visualización de arrays de cámaras lineales (Sección 2.1.3).

Una vez se dispone de la posición horizontal de las Oculus Rift, se calcula el índice de la vista correspondiente del array de cámaras.

$$Nº\ vista_{izq} = Round\left(\frac{posicion.x}{d\_conmutacion} + \frac{n\_camaras}{2}\right)$$

Donde:

- *posicion.x*: coordenada horizontal de la posición extraída
- *d_conmutacion:* de ahora en adelante "distancia de conmutación entre vistas", es la distancia que debe mover el usuario la cabeza para modificar el punto de vista (Sección Parametrización)
- *n_camaras:* número de cámaras del array

Dividiendo la posición por la distancia de conmutación entre vistas permite obtener el índice de la vista correspondiente a cada posición. Adicionalmente, y con el fin de asociar la posición central (0 cm) del usuario a la vista central del array, se suma la mitad del número de cámaras presentes en el array.

No obstante, la posición del usuario no debe nunca sobrepasar los límites cubiertos por el array de cámaras de entrada. Por tanto, es necesario acotar tanto inferiormente como superiormente el índice calculado, teniendo en cuenta las vistas adicionales correspondientes a la distancia interocular y el hecho de que la numeración de vistas esté diseñada para empezar en 0:

$$Nº\ vista_{izq-MIN} = 0$$

$$Nº\ vista_{izq-MAX} = n\_camaras - d\_interocular - 1$$

Una vez obtenido el índice de la vista correspondiente al ojo izquierdo, sólo falta sumarle el número de saltos de vista correspondiente a la distancia interocular (que introduciremos en la Sección Parametrización), para obtener el índice de la vista correspondiente al ojo derecho:

$$Nº\ vista_{der} = Nº\ vista_{izq} + d\_interocular$$

Adicionalmente, y con el fin de facilitar la realización de pruebas sin HMD, se han habilitado otras dos teclas que permiten simular el movimiento de la cabeza mediante input de teclado:

- Tecla 'A': mover las Oculus Rift hacia la izquierda
- Tecla 'D': mover las Oculus Rift hacia la derecha

3.2.2.2. Reproducción en segundo plano

Para proporcionar al sistema la mayor fluidez posible entre cambios de punto de vista, como ya hemos explicado en la Sección 3.1, se encuentran reproduciéndose un conjunto de vistas en segundo plano en torno a la posición del usuario en cada momento (Figura 21).

Figura 21 - Concepto de reproducción en segundo plano (modo normal, 8 vistas)

El hecho de tener estas vistas en reproducción en segundo plano reduce considerablemente el tiempo necesario para cambiar el punto de vista, al ser el tiempo de texturización a pantallas ínfimo en comparación al tiempo necesario para reproducir una vista. Adicionalmente, el tener este conjunto de vistas en reproducción en segundo plano permite también poder ir sincronizándolas con el tiempo global del cuadro (como veremos en la Sección 3.2.2.3), teniendo las vistas listas en todo momento para ser texturizadas (como veremos en la Sección 3.2.2.4).

Así mismo, también se reproducirán en segundo plano las vistas correspondientes a la distancia interocular (entre la vista izquierda y derecha), adicionalmente al número de vistas en reproducción escogido.

Esta operación se repite en cada bucle de operación tras la extracción de la posición de la cabeza, descartando las vistas que pasan a estar fuera del conjunto y reproduciendo las que van formando parte del conjunto de vistas en reproducción en segundo plano. Esta implementación proporcionará fluidez y por tanto mayor sensación de inmersión al usuario, siempre y cuando éste mueva la cabeza a un ritmo adecuado, sin salirse así del conjunto de vistas en reproducción en segundo plano en cada momento.

Durante el desarrollo del proyecto se han implementado distintos modos de reproducción en segundo plano, empezando por el desarrollo de una serie de modos fijos que finalmente dieron lugar a la creación de un modo adaptativo, capaz de adaptarse al movimiento del usuario.

El modo adaptativo destaca frente al resto, al ser capaz de ajustar el número de vistas en reproducción en cada sentido a la posición del usuario extraída en cada momento. Si un usuario se acerca a las primeras o últimas cámaras del array de cámaras, el modo adaptativo es capaz de reproducir también el número de vistas que se pierdan más allá de los límites del array (Sección 3.1).

Los demás modos, denominados modos fijos (normal, *on my way* y *on the other way*), fueron el origen del modo adaptativo y se mantienen por si resultasen interesantes en futuras pruebas subjetivas que comparen el comportamiento del usuario frente a distintos modos. El modo adaptativo trata de adaptarse a la utilización normal de un usuario, y será por tanto el más adecuado en la gran mayoría de casos.

- MODOS FIJOS:

  - Modo normal: Es el modo básico (Figura 21) a partir del cual se desarrolló el modo adaptativo. Se reproducen en segundo plano, adicionalmente a las vistas presentes entre los dos ojos, el mismo número de vistas en ambas direcciones (tomando la posición de las gafas como origen) independientemente del sentido de movimiento de la cabeza del usuario y de la posición.
  - Modo *on my way*: Este modo reproduce el 75% de las vistas totales a reproducir en segundo plano en el sentido en el que el usuario esté moviendo la cabeza; mientras que sólo reproduce el 25% en el sentido opuesto al movimiento (Figura 22). Este modo prioriza por tanto el sentido del movimiento del usuario.

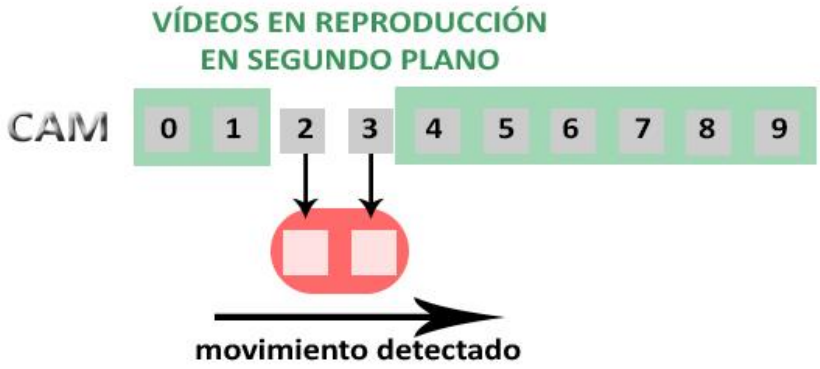


Figura 22 - Modo de reproducción "on my way" (8 vistas)

  - Modo *on the other way*: Este modo prioriza el sentido opuesto al del movimiento del usuario. Se reproducirán en segundo plano el 25% de las vistas en el sentido del movimiento y el 75% en el sentido opuesto al movimiento que está trazando la cabeza del usuario. (Figura 23).

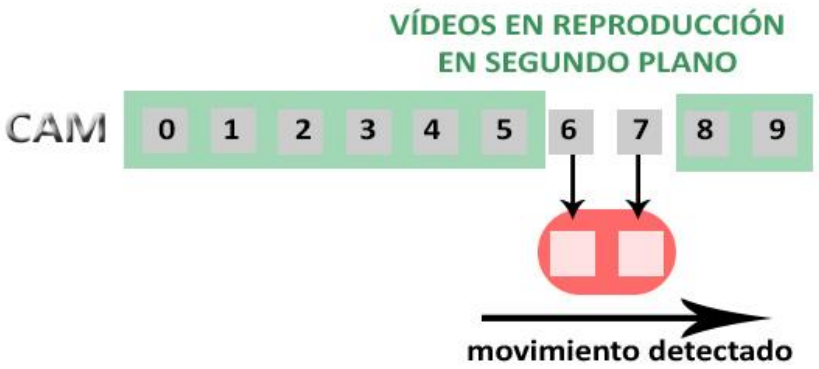


Figura 23 - Modo de reproducción "on the other way" (8 vistas)

La implementación actual de los modos *on my way* y *on the other way* trabaja dividiendo la cantidad de vistas a reproducir en proporciones 25%-75% (o 75%-25%). No obstante, este ratio es fácilmente modificable y podría reajustarse en un futuro.

Para la obtención del sentido de movimiento del usuario, se ha realizado un filtrado temporal del movimiento. Si durante tres instantes seguidos (tres actualizaciones de cuadro), hay movimiento en un mismo sentido, se considera que el usuario está moviendo la cabeza en ese sentido. Si, en cambio, no se detecta movimiento en el mismo sentido en los tres instantes, no se considera que el usuario esté moviendo la cabeza en ese sentido, se mantiene el modo normal y se sigue buscando el sentido del movimiento desde la próxima actualización de cuadro. En el momento en el que se encuentre un sentido de movimiento el modo *on my way* (u *on the other way)* se activará.

La utilización de este filtrado temporal permitirá, adicionalmente, evitar que los posibles errores en el posicionamiento de las Oculus Rift (errores de precisión en la medida) afecten a la obtención del sentido de movimiento del usuario.

- MODO ADAPTATIVO:

En este modo de reproducción en segundo plano la distribución de vistas a la derecha e izquierda se adapta a la posición del usuario, siendo uniforme para las posiciones centrales y variando en las posiciones cercanas a los límites del array. En este modo se distinguen por tanto tres situaciones:

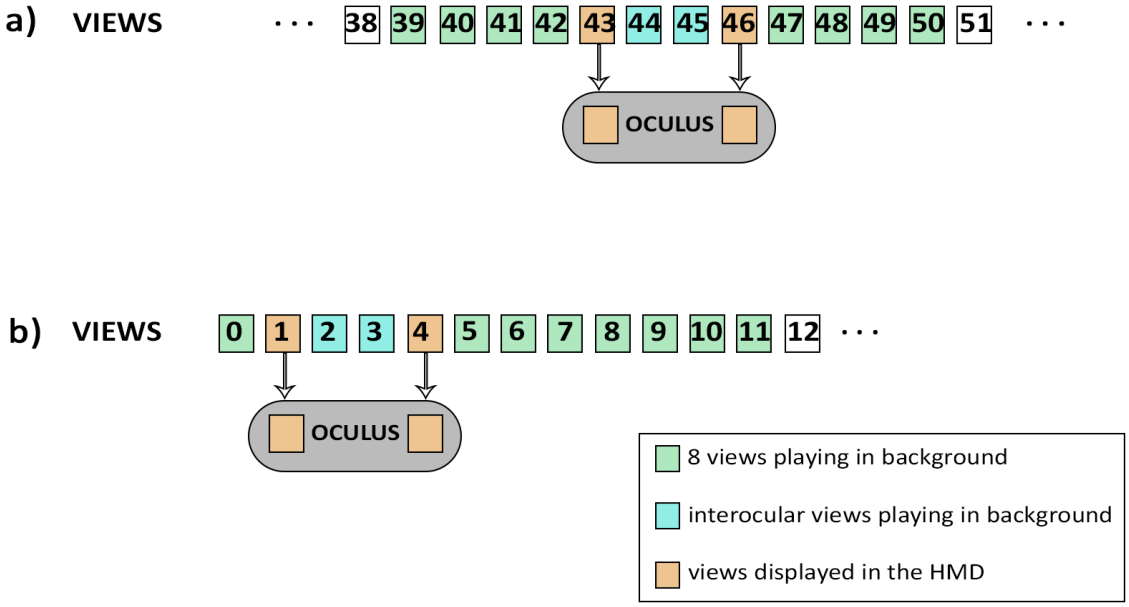


Figura 24 - Conjunto de vistas en reproducción con modo adaptativo (8 vistas)
a) para un usuario situado en la posición central
b) para un usuario cercano al límite izquierdo

- El usuario se encuentra próximo a la posición central del array de cámaras. El modo funciona como el modo normal, reproduciendo el mismo número de vistas en ambos sentidos (*a)* en Figura 24).
- El usuario se encuentra cerca del límite izquierdo del array de cámaras. En otras palabras, a la izquierda del usuario quedan menos vistas disponibles que las asociadas al conjunto a reproducir en segundo plano. En este caso, el modo adaptativo cambia el ratio 50%-50% para pasar a reproducir hacia la derecha las vistas que se perderían más allá del límite izquierdo del array (*b)* en Figura 24).
- El usuario se encuentra cerca del límite derecho del array de cámaras. El funcionamiento es análogo al descrito para el límite izquierdo.

Adicionalmente, se ha habilitado el cambio entre los distintos modos de reproducción en segundo plano durante la ejecución del programa mediante el uso del teclado:

- Tecla 'F1': Modo Adaptativo
- Tecla 'F2': Modo Normal
- Tecla 'F3': Modo *On My Way*
- Tecla 'F4': Modo *On The Other Way*

La implementación de estos modos de reproducción en segundo plano no es más que un ejemplo de lo que se podría conseguir en desarrollos futuros, tras extraer un log de trayectorias que permita analizar las trayectorias de los usuarios y así poder crear nuevos modos de reproducción en segundo plano adaptados al movimiento habitual de los usuarios (Sección 5.2).

3.2.2.3. Sincronización entre vistas

Este bloque es el responsable de sincronizar todas las vistas incluidas en el conjunto en reproducción en segundo plano. Es importante mantener las vistas del conjunto sincronizadas en todo momento, puesto que se podría obtener un deterioro grave de la calidad de experiencia por la desincronización entre vistas texturizadas.

Para llevar a cabo sincronización del sistema, se utiliza un tiempo absoluto de cuadro que será el responsable de mantener sincronizadas todas las vistas. Este tiempo representará el instante del vídeo inicial que se está reproduciendo en cada momento, permitiendo así tener una referencia de en qué punto se deben reproducir los vídeos.

La fase de sincronización consta de varios pasos:

1. Se verifica si todas las vistas dentro del rango están correctamente sincronizadas. Para ello, se compara el tiempo de reproducción de cada vídeo con el tiempo absoluto de cuadro.
2. Se asegura que las vistas desincronizadas se encuentran en reproducción.
3. Adelanta las vistas desincronizadas para que sigan reproduciéndose en el tiempo absoluto de cuadro.
4. Se actualiza el tiempo absoluto de cuadro:
    a. Sumando el tiempo de lectura de cada cuadro:

$$Tiempo\ abs\ cuadro_{en\ t=i} = Tiembo\ abs\ cuadro_{en\ t=i-1} + Tiempo\ último\ cuadro$$

    Donde *tiempo último cuadro* representa el tiempo que ha llevado la última ejecución del bucle de operación.
    b. Reinicializando el tiempo absolto de cuadro cuando el vídeo se haya terminado y deba comenzar de nuevo (*loop activado)*.

Para adelantar las vistas al tiempo absoluto se ha utilizado la función *Seek*, por lo que se ha necesitado utilizar un paquete de Unity [31] que añade esta funcionalidad (*.Seek()*) a la función de texturización de vídeo básica de Unity (*MovieTexture*). La utilización de este paquete obliga a adaptar el sistema para que se reproduzca en un navegador con WebGL [32] (como ya mencionamos en la Sección 2.4.2 y veremos en en apéndice 7.1).

Adicionalmente, se ha activado una tecla para casos de emergencia que fuerce la sincronización inmediata:

- Tecla ‘S’: Forzar sincronización inmediata

3.2.2.4. Texturización

Este último bloque texturiza las vistas correspondientes a la nueva posición extraída, que normalmente ya se encuentran en reproducción sincronizada en segundo plano, en las dos pantallas de la escena 3D. Este bloque se activará cuando sea necesario texturizar una nueva vista (un nuevo vídeo), no considerándose en él la texturización de los distintos cuadros de un mismo vídeo, lo cual maneja automáticamente Unity.

Se texturizará nuevo contenido en las pantallas si:

- El índice de vista correspondiente a la posición actual es distinto del índice de la vista correspondiente a la posición extraída en el cuadro anterior.
- Se trata del primer cuadro de la ejecución y se necesita una textura inicial para las pantallas.
- La tecla 'S' ha sido presionada.

Si se cumple una de las situaciones anteriores, se texturizan las nuevas vistas en cada una de las pantallas, cambiando la textura asociada a cada uno de los objetos 'pantalla' de la escena 3D.

3.2.3. Parametrización

Un gran objetivo del proyecto es conseguir que este sistema sea lo más parametrizable posible, permitiendo al usuario que lo utilice configurar todos los parámetros útiles de cara a la utilización del sistema con diferentes secuencias multivista o a probar distintas configuraciones para una misma secuencia.

Los parámetros serán introducidos por el usuario mediante un menú de configuración (interfaz de usuario) que aparecerá nada más abrir la aplicación y que describimos en el manual de usuario del apéndice Manual de usuario

Los distintos parámetros configurables en nuestro sistema son:

- **Formato de nombre de los vídeos de entrada:** es el nombre de los ficheros de los vídeos de entrada (Figura 18).

- **Cantidad de vídeos del array:** representa el número total de vistas de la secuencia importada y se ha utilizado para calcular los límites de posicionamiento en el sistema (Sección 3.2.2.1).

- **Distancia interocular (en número de saltos de vista):** representa la separación entre las vistas visualizadas en el ojo izquierdo y el ojo derecho. Por comodidad, se representa en número de saltos de vista (en lugar de unidad métrica).
- **Modo de reproducción en segundo plano:** este parámetro permite al usuario escoger cuál de los cuatro modos de reproducción en segundo plano (descritos anteriormente en la Sección 3.2.2.2) desea utilizar inicialmente.

- **Número total de vistas en reproducción en segundo plano:** representa el número total de vistas en reproducción en segundo plano. El valor introducido corresponde a la suma las vistas que se encuentran en reproducción hacia posiciones situadas a la izquierda y derecha del usuario, sin incluir las vistas correspondientes a la distancia interocular (Figura 17).

- **Resolución de los vídeos de entrada:** obtiene la resolución de los vídeos de entrada, con el fin de poder mantener la relación de aspecto a la hora de crear las pantallas de escena 3D en las que se texturizarán las vistas (Sección 3.2.1).

- **Distancia de conmutación entre vistas (en mm):** es la distancia que el usuario debe mover la cabeza hacia la izquierda o derecha para pasar a la vista anterior o siguiente, respectivamente. Este parámetro posibilita la obtención del índice de vista

correspondiente a la posición extraída (Sección 3.2.2.1) en cada ejecución del bucle de operación.

- **Rotación (Modo de simulación de pantalla):** el usuario tiene la posibilidad de desactivar el giroscopio de las Oculus Rift con el fin de hacer que las pantallas "sigan" la orientación de su cabeza, o mantener la rotación activada, fijando así su posición en la escena 3D y simulando el comportamiento de una pantalla.

- **Distancia a las pantallas:** es la distancia entre las pantallas y la cámara que representa las Oculus Rift en la escena 3D. Se mide en relación al alto (*height)* de las pantallas (resolución vertical de la secuencia). Este parámetro sólo es ajustable durante la ejecución.

4. PRUEBAS EXPERIMENTALES

En esta sección ponemos a prueba el sistema implementado, analizando los distintos parámetros configurables mediante una serie de pruebas experimentales, que permitirán localizar los valores más adecuados de dichos parámetros para las distintas secuencias SMV utilizadas.

4.1. Escenario de las pruebas

La aplicación final del proyecto, con el fin de volverla funcional para distintas secuencias SMV de entrada, y al estar principalmente destinada a la realización de pruebas subjetivas de calidad de vídeo 3D multivista, presenta una serie de parámetros configurables por el usuario que ya han sido descritos (Sección 3.2.3), y que son los que pasaremos a analizar a continuación. El principal objetivo de estas pruebas es el encontrar los valores de los distintos parámetros configurables del sistema que optimicen la experiencia de usuario para cada una de las secuencias y configuraciones.

4.1.1. Características técnicas

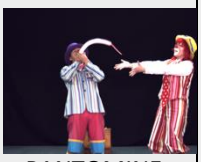
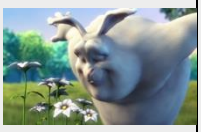
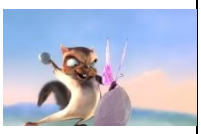
Para las pruebas se ha utilizado un PC del laboratorio del Grupo de Tratamiento de Imágenes de la ETSIT. Los límites de funcionamiento del sistema obtenidos pueden variar de un ordenador a otro y dependerán de las características técnicas que éste presente. Por eso, se ha tomado dicho ordenador como referencia, el cual presenta las siguientes características:

- Intel® Core™ i7-4790 CPU @ 3.60 GHz
- RAM 16GB
- 64-bit OS
- NVIDIA GeForce GTX 970 (4095 MB)

4.1.2. Secuencias utilizadas

Estas pruebas se han realizado para las secuencias de vídeo SMV de MPEG [33][34], utilizadas a lo largo de todo el proyecto. Estas secuencias, consideradas como SMV (Sección 2.1.4), son las más adecuadas para el sistema implementado, al incluir estas un gran número de vistas disponibles. En la Tabla 1 se describen sus principales características:

Tabla 1 - Características de las secuencias de vídeo utilizadas de ejemplo

Nombre	Tipo de Array	Tipo de Contenido	Número de Cámaras	Resolución Espacial	Separación entre Cámaras	Resolución Temporal
 PANTOMINE	Lineal	Real	80	1280x960	50 mm	29 fps
 CHAMPAGNE	Lineal	Real	80	1280x960	50 mm	29 fps
 FLOWERS	Lineal	Sintético	91	1280x768	0.0091 Blender Units[1]	24 fps
 BUTTERFLY	Lineal	Sintético	91	1280x768	0.0086 Blender Units	24 fps
FLOWERS ARC	Arco	Sintético	91	1280x768	0.5º	24 fps
BUTTERFLY ARC	Arco	Sintético	91	1280x768	0.5º	24 fps

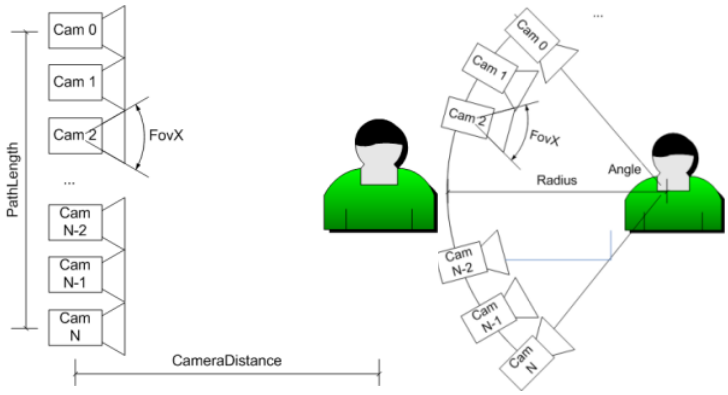


Figura 25 - Ejemplo de array lineal y array en forma de arco [34]

En la tabla anterior observamos dos tipos de array de cámaras: los arrays lineales y los arrays en forma de arco. Aunque el sistema está orientado a movimientos horizontales del usuario, en las configuraciones en arco la distancia del conjunto de cámaras al plano de convergencia es suficiente para aproximar el movimiento a través de las cámaras como un movimiento lineal (Figura 25).

[1] *Las secuencias de contenido sintético han sido creadas por ordenador y no existe una distancia real de separación entre cámaras, por lo que se ha decidido expresarlo en las unidades del programa utilizado.*

También se identifica otro parámetro distintivo entre vídeos, el tipo de contenido del vídeo en cuestión. Distinguimos entre dos tipos de contenido: vídeo real y vídeo sintético. En un vídeo real se capturan escenas del mundo real, mientras que en un vídeo sintético estas escenas han sido creadas por ordenador.

4.2. Parámetros y límites

Durante la realización de estas pruebas, y salvo que se indique lo contrario, se han mantenido constantes los siguientes parámetros:

- Distancia entre la cámara y las pantallas de texturización fijada a 3 veces la altura del vídeo de entrada (*height*), por ser la distancia de *3H* una distancia comúnmente utilizada para la realización de pruebas subjetivas con pantallas físicas.
- Modo adaptativo de reproducción en segundo plano, al ser el más adecuado para un uso normal del sistema.
- Rotación activada, por ser el modo que simula una pantalla corriente.

4.2.1. Resolución de los vídeos de entrada

Las secuencias de entrada tienen originalmente las siguientes resoluciones: 1280x768 para las secuencias sintéticas y 1280x960 para las secuencias reales.

A la hora de seleccionar la resolución adecuada para el sistema, es importante conocer la resolución disponible en el display para cada una de las vistas. Así mismo, las pantallas en las que se texturizan las vistas no ocupan la totalidad de la parte del display que les ha sido asignada. Tomando como referencia la distancia de *3*height* mencionada anteriormente, las pantallas no ocupan más del 80-85% de la resolución horizontal del display asociada a cada ojo. Conociendo la resolución de las Oculus Rift DK2 (Sección 2.2.2), la cual es de 960x1080 por ojo, podemos deducir que la resolución horizontal máxima ocupada por las pantallas en esa posición será de aproximadamente 780-820 píxeles. Por tanto, el uso de resoluciones con componente horizontal superior a este valor no presentaría a priori ventajas perceptibles de calidad de imagen.

A continuación, mostramos los resultados obtenidos para el máximo número de vistas soportados en reproducción en segundo plano para las distintas resoluciones estudiadas. Durante estas pruebas se mantuvieron los demás parámetros fijos. No se entrará en detalle en la explicación de los resultados aquí obtenidos (Tabla 2 y

Tabla 3) al estar estos explicados en las próximas secciones (Sección 4.2.3); y nos centraremos en analizar la limitación que provoca el uso de una u otra resolución.

Tabla 2 - Número máximo de vistas en reproducción en segundo plano para distintas resoluciones de las secuencias sintéticas

	Flowers (arco y lineal)	**Butterfly (arco y lineal)**
1280x768	14	11
640x384	40	36

Tabla 3 - Número máximo de vistas en reproducción en segundo plano para distintas resoluciones de las secuencias reales

	Champagne	**Pantomine**
1280x960	14	11
640x480	40	36

Reduciendo las resoluciones horizontales y verticales en ½, no se han observado grandes cambios en la calidad de la imagen, frente a la gran reducción de tamaño de los ficheros de entrada. Por estos motivos y tras observar una clara mejoría del rendimiento del sistema, permitiendo a su vez optimizar los valores de los distintos parámetros del sistema y mejorando drásticamente la experiencia al utilizar el sistema, se ha decidido utilizar las resoluciones reducidas SD.

Los valores adecuados encontrados para la resolución de los vídeos de las distintas secuencias son por tanto (Tabla 4):

Tabla 4 - Resoluciones más adecuadas para las secuencias de ejemplo

Flowers (arco y lineal)	**Butterfly (arco y lineal)**	**Champagne**	**Pantomine**
640x384	640x384	640x480	640x480

El uso de las resoluciones HD ha presentado peores resultados debido a la carga computacional requerida. Eso causa que en cada actualización de cuadro, el número de vistas en reproducción en segundo plano deba limitarse, limitando a su vez considerablemente la velocidad a la que el usuario puede desplazar la cabeza. Esto puede dar lugar a la aparición de desincronizaciones si el movimiento de cabeza no se hace lo suficientemente lento como para dar tiempo a actualizar las vistas en reproducción en segundo plano.

Para obtener buena fluidez en la ejecución del sistema y poder optimizar los distintos parámetros del sistema, se recomienda el uso de resoluciones no muy superiores a las indicadas en la Tabla 4. La adaptación del sistema para su correcto funcionamiento con resoluciones HD queda pendiente como línea futura de desarrollo, de cara a poder adaptar el sistema a nuevos HMDs de mayor resolución.

4.2.2. Distancia interocular

Durante la realización de estas pruebas se mantuvieron, adicionalmente a los ya destacados, los siguientes parámetros fijos:

- Resoluciones de los vídeos de entrada indicadas en el apartado anterior (Tabla 4).
- Para evaluar el rango de distancias interoculares adecuado para cada secuencia, se ha utilizado un par fijo de vistas. Para mantener este par fijo de vistas ha sido necesario disminuir al mínimo el número de vistas en reproducción en segundo plano y aumentar

considerablemente la distancia de conmutación entre estas, impidiendo así el cambio entre vistas:

- Únicamente 2 vistas en reproducción en segundo plano.
- Distancia de conmutación entre vistas excesivamente grande fijada, por ejemplo, a 40 cm.

Esta "distancia" interocular, representada realmente en saltos entre vistas, corresponde al número de saltos que separan las dos vistas (vista izquierda y derecha de las Oculus Rift). Otra forma de ver este parámetro es como la diferencia entre los índices correspondientes a la vista derecha y a la vista izquierda (Figura 26).

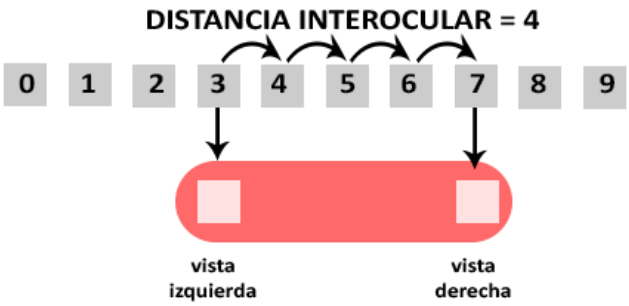


Figura 26 – Ejemplo de distancia interocular en número de saltos de cámara

Normalmente, en la producción de contenido estéreo, la distancia interocular suele tomar el valor medio de la distancia entre los dos ojos de un ser humano, que corresponde a 6-7cm. No obstante, a la hora de visualizar el contenido, la sensación de profundidad puede verse afectada por múltiples parámetros como la distancia al plano de convergencia, la distancia al display, la distancia focal de las cámaras, el contenido de la escena… Todo esto, hace que dependiendo del caso, el sistema pueda dar mejores resultados subjetivos para distancias interoculares mayores o menores que el valor teórico humano.

Durante estas pruebas buscamos el rango de valores para la distancia interocular que optimice la sensación de confort y profundidad 3D. Valores excesivamente bajos de este parámetro causarían la pérdida de la percepción de profundidad y valores excesivamente altos podrían deteriorar el confort 3D.

Los rangos de valores adecuados encontrados para el número de saltos de vista que determina la distancia interocular son (Tabla 5):

Tabla 5 - Distancias interoculares adecuadas para las secuencias de ejemplo (en número de saltos entre vistas)

	Flowers (arco y lineal)	**Butterfly (arco y lineal)**	**Champagne**	**Pantomine**
Rango adecuado	3-7	5-14	1-3	1-5
Valor más confortable	5	9	2	2

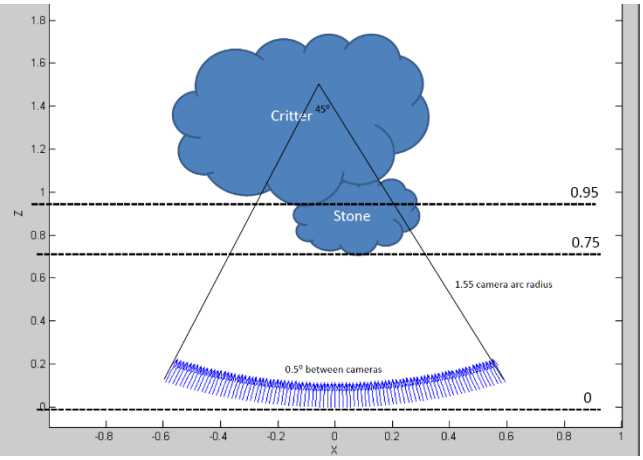


Figura 27 - Esquema del array en arco utilizado para la secuencia Butterfly [34]

Como podemos observar en la Tabla 5, en el caso de las secuencias sintéticas, se han obtenido unos rangos adecuados de distancias interoculares idénticos para las dos configuraciones posibles (array lineal de cámaras y array en arco). Esto es debido a la poca separación entre las distintas cámaras (0.5º como vemos en la Tabla 1), que da lugar a la formación de un arco (Figura 27) corto con respecto al radio que se asimila al resultado obtenido con el array de cámaras lineal.

Esto, unido al hecho de que la similitud de estos array con los lineales se acentúa en las cámaras centrales (en las cuales nos centramos para la realización de estas pruebas), al estar estas situadas a distancias muy próximas a las utilizadas en el array lineal, son los dos principales aspectos que causan que se obtengan distancias interoculares idénticas para las dos configuraciones. No obstante, esto no es lo habitual y puede no repetirse para otras arrays en arco con más cámaras, con más separación angular entre cámaras…

4.2.3. Distancia de conmutación entre vistas

Durante la realización de estas pruebas se mantuvieron los siguientes parámetros fijos, además de los ya mencionados:

- Resoluciones adecuadas de los vídeos de entrada (Tabla 4).
- Distancia interocular fijada a los valores más confortables encontrados en el apartado anterior (Tabla 5).
- Número de vistas en reproducción en segundo plano fijado a 20, suficientemente alto para poder visualizar las transiciones entre estas (sin afectar considerablemente al rendimiento del programa).

Este es uno de los parámetros más relevantes del sistema, y dictamina la distancia que debe el usuario mover la cabeza hacia la izquierda/derecha para pasar de una vista a la anterior/siguiente. La percepción será mejor cuanto más se aproxime el movimiento de cabeza del usuario al cambio de punto de vista entre dos cámaras contiguas del array.

En el caso de las secuencias sintéticas utilizadas de ejemplo, la limitación ocasionada por la distancia entre las cámaras del array de captura no es importante, al estar generadas por ordenador y al tener distancias entre cámaras virtuales. No obstante, en el caso de las secuencias reales, observamos en la tabla de presentación de las secuencias (Tabla 1) que estas han sido capturadas por cámaras distantes 5cm; dato a tener en cuenta para la obtención de la mejor percepción posible.

Durante estas pruebas se busca ir reduciendo el valor de la distancia de conmutación entre vistas hasta encontrar los valores óptimos que faciliten una buena suavidad a la hora de realizarse la transición entre dos puntos de vistas, teniendo cuidado a su vez de no tomar valores demasiados

pequeños que causen que pequeños movimientos de cabeza se reflejen en que los movimientos de los objetos capturados en la escena sean excesivamente grandes.

Así mismo, existen otros dos factores que juntos pueden acotar inferiormente esta distancia de conmutación, que son:

- El error de medida de la cámara de posicionamiento de las Oculus Rift, teóricamente despreciable para usuarios situados entre 0.5m y 2.5m de la cámara.
- El posible movimiento involuntario del usuario al tratar de estar completamente inmóvil, que puede alcanzar 5mm de error.

Estos dos factores causan que se observen transiciones involuntarias para distancias de conmutación inferiores a 5mm, independientemente de la secuencia de entrada.

El rango de movimiento que el usuario puede realizar en un refresco de cuadro (mitad hacia cada lado) viene determinado por dos parámetros: la distancia de conmutación ($d\_conmutacion$) y el número de vistas en reproducción en segundo plano ($n\_vistas2p$). Es importante entender que si la distancia de conmutación es demasiado pequeña, el usuario puede realizar movimientos de cabeza que terminen en vistas no incluidas en el conjunto de vistas en reproducción en segundo, dando lugar a desincronizaciones mientras esta nueva vista es puesta en reproducción y posteriormente sincronizada, deteriorando así la experiencia del usuario y la sensación de inmersión.

Así pues, este parámetro va directamente ligado con el parámetro referente al *número de vistas en reproducción en segundo plano*, por lo que habrá que encontrar un buen equilibrio entre estos dos valores. La distancia máxima que puede mover un usuario la cabeza entre cuadros consecutivos viene determinada por la ecuación:

$$Distancia\ max_{izquierda/derecha} = \frac{n\_vistas2p\ \times d\_conmutacion}{2}$$

Los rangos de valores adecuados y valores más confortables encontrados para las distintas secuencias que proporcionan suavidad en la transición entre vistas sin llegar a empeorar la percepción de la escena 3D (Tabla 6) son:

Tabla 6 - Distancia de conmutación entre vistas más adecuada para secuencias de ejemplo

	Flowers (arco y lineal)	**Butterfly (arco y lineal)**	**Champagne**	**Pantomine**
Rango adecuado	5mm - 15mm	5mm - 15mm	20mm - 50mm	30mm - 60mm
Valor más confortable	10mm	10mm	30mm	40mm

Para las secuencias de contenido sintético, se han obtenido los resultados más confortables para distancias de conmutación de 10mm, lo cual permite obtener una gran fluidez en la transición entre vistas y deja margen para no verse afectado por los posibles errores de medida. En cambio, este valor se ve incrementado para las secuencias de contenido real, lo cual se debe principalmente a la elección de un valor no muy inferior a la separación original entre cámaras (Tabla 1), evitando que así pequeños movimientos de cabeza se vean reflejados en grandes desplazamientos de los objetos capturados en la escena.

4.2.4. Número de vistas en reproducción en segundo plano

Durante la realización de estas pruebas se mantuvieron los siguientes parámetros fijos, adicionalmente a los ya destacados:

- Resoluciones adecuadas de los vídeos de entrada (Tabla 4).
- Distancia interocular fijada a los valores óptimos (Tabla 5)
- Distancia de conmutación entre vistas fijada al valor más adecuado indicado en el apartado anterior (Tabla 6)

Este parámetro, junto al de la distancia de conmutación (Sección 4.2.3), son los dos que más afectan tanto al rendimiento del programa como a la percepción final de 3D. El número total de vistas aquí indicado representa la cantidad de vistas laterales que estarán reproduciéndose en segundo plano en cada momento para poder ser texturizadas cuando usuario sitúe la cabeza en la posición correspondiente a dicha vista (Figura 17).

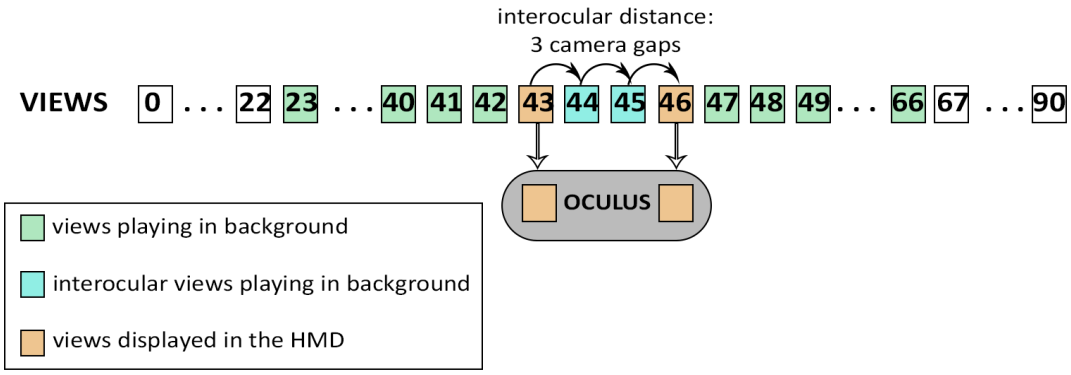


Figura 17 - Ejemplo de vistas en reproducción en segundo plano para un usuario situado en la posición

Se desea encontrar el número óptimo de vistas a tener en reproducción en segundo plano para una secuencia dada, siendo este valor constante durante la ejecución del programa. Cuanto más alto sea este valor, más vistas habrá preparadas y más distancia en menos tiempo podrá recorrer el usuario sin que se desincronicen las vistas de los dos ojos. Esto último ocurrirá cuando el usuario pase a estar en una región donde no haya vistas preparadas (en reproducción síncrona en segundo plano) y el sistema deba preparar la nueva vista, lo cual no es instantáneo y puede dar lugar a estas desincronizaciones.

En una situación ideal, se reproducirían todas las vistas restantes del array y el usuario podría mover la cabeza a su gusto y seguiría visualizando dos vistas sincronizadas en todo momento. No obstante, esto será raramente posible, y se ha de tener especial cuidado con la elección del valor para este parámetro, puesto que la operación de reproducción en segundo plano requiere de una gran capacidad de procesado. Adicionalmente a lo ya mencionado, el valor de este parámetro también se verá directamente condicionado por el tamaño de los ficheros de video de entrada; cuanto más grande sea el tamaño de estos ficheros más capacidad de procesado requerirá su reproducción en segundo plano.

Cabe recordar que, en caso de tener una distancia interocular superior a una vista, las vistas comprendidas entre los dos ojos también se estarán reproduciendo en segundo plano. Eso hace

que, si se desea aumentar el número de vistas entre los ojos, se deba disminuir el número total de vistas en reproducción en segundo plano (parámetro aquí descrito), para así poder seguir teniendo un buen rendimiento del programa (como ya vimos en la Figura 17).

Así pues, se han obtenido los siguientes resultados para el número de vistas en reproducción en segundo plano:

Tabla 7 - Número de vistas en reproducción en segundo plano para las secuencias de ejemplo

	Flowers (arco y lineal)	**Butterfly (arco y lineal)**	**Champagne**	**Pantomine**
Rango adecuado	30 - 50	26 - 46	20 - 40	10 - 20
Valor más confortable	40	36	30	14

Este valor confortable (Tabla 7) variará considerablemente al modificar la distancia de conmutación ya mencionada y también dependerá de los hábitos de movimiento de cabeza que tenga el usuario que utilice el sistema.

En este caso se han observado resultados peores para la secuencia *Pantomine* al estar ésta compuesta por archivos de mayor tamaño tras la compresión (Sección 7.2), y la cual da lugar a ciertas desincronizaciones si se escoge un número de vistas en reproducción en segundo plano superior al límite marcado en su rango de valores adecuados.

El programa funciona correctamente para las secuencias de vídeo utilizadas de ejemplo (con las resoluciones sacadas de la Sección 4.2.1), lo cual permite aumentar mucho este número para las secuencias sintéticas (Tabla 7), alcanzando valores de prácticamente la mitad de la cantidad de cámaras totales del array en el caso de la secuencia Flowers. No obstante, como vimos en la Tabla 2 y

Tabla 3, este valor deberá reducirse al utilizar resoluciones mayores, como las resoluciones originales de las secuencias de ejemplo (Tabla 1).

4.2.5. Modo de reproducción en segundo plano

Durante la realización de estas pruebas se mantuvieron los siguientes parámetros fijos, adicionalmente a los ya destacados:

- Resoluciones adecuadas de los vídeos de entrada (Tabla 4).
- Distancia interocular fijada a los valores óptimos encontrados (Tabla 5).
- Distancia de conmutación entre vistas fijada al valor más adecuado (Tabla 6).
- Número de vistas en reproducción en segundo plano fijada al valor más confortable (Tabla 7).

Como ya se ha descrito en el punto correspondiente a los modos de reproducción en segundo plano (Sección 3.2.2.2), la cantidad de vistas en reproducción no varía en función del modo escogido para posiciones centrales; por tanto, la elección de un modo u otro no debería causar fluctuaciones en el rendimiento del programa.

El objetivo de estas pruebas ha sido por tanto la evaluación de los diferentes modos con el fin de poder concluir en qué situaciones sería adecuado utilizar uno u otro modo. La elección de modo de reproducción en segundo plano no dependerá de la secuencia utilizada ni de ningún otro parámetro de los ya descritos en esta sección. Por tanto, durante estas pruebas se ha utilizado únicamente una secuencia (*Flowers*) y nos hemos centrado en estudiar cómo mueve la cabeza un usuario medio durante la ejecución del programa y como esto influye en la elección de uno u otro modo. Los resultados aquí descritos deberán corroborarse en un futuro mediante la realización de más pruebas subjetivas.

- Modos Fijos:

  - El modo normal (Sección 3.2.2.2), el cual reproduce el mismo número de vistas a la izquierda y a la derecha de la posición actual, presenta mejores resultados visuales para movimientos de cabeza en torno a una posición central; permitiendo "idas" y "vueltas" de cabeza sin apenas probabilidades de desincronización. Este método parece ser a su vez el más adecuado de los modos fijos para este tipo de pruebas subjetivas; puesto que el usuario que porta la gafas pierde la visión de lo que tiene a su alrededor y tiende a mover la cabeza en torno a una posición inicial para evitar chocar contra objetos que tenga alrededor. Así mismo, también se ha observado que el usuario tiende a hacer este tipo de movimiento para "pasear" la vista alrededor del objeto de interés del vídeo, aumentando así la percepción 3D de dicho objeto.

  - El modo *on my way* que reproduce el 75% del total de las vistas a reproducir en el sentido del movimiento de cabeza y el 25% en el sentido opuesta, resulta más adecuado para situaciones de "barrido" de las vistas del array. Es decir, será más adecuado para movimientos de cabeza continuos en un sentido; óptimo si se desea recorrer el array. Presenta a su vez peores resultados que el modo normal para movimientos oscilatorios en torno a un punto, puesto que estos conllevan movimientos de cabeza en ambos sentidos.

  - El modo *on the other way*, que por su parte, reproduce el 25% del total de vistas a reproducir en el sentido del movimiento de cabeza y el 75% en el sentido opuesto, resulta más adecuado para las posiciones cercanas a los límites del array (cercanas a la primera o última vista del array). Esto se debe a que este modo, al llegar a estas posiciones, pasa a reproducir la mayor parte de las vistas en el sentido opuesto al límite del array, sentido al tenderá a mover la cabeza una vez llegue al límite del array. Esto optimizará el programa en este tipo de situaciones, en lugar de "perderse" posibles vistas que intenten reproducirse más allá de los límites del array.

- Modo adaptativo:

  Recordamos que este modo se comporta de forma similar al modo normal, salvo cuando el usuario se acerca a los límites del array, momento en el que el sistema empezará a reproducir en el sentido contrario las vistas que "pierda" más allá del límite. Se ha comprobado que este modo presenta las ventajas ya descritas de los modos anteriores combinadas, sin la desventaja que presentan los demás modos al ser fijos para todas las posiciones del usuario. Por tanto, parece que este modo es el más adecuado para un uso

normal en pruebas subjetivas de calidad de experiencia, puesto que es el que más se adapta al movimiento realizado por el usuario durante las pruebas experimentales.

4.2.6. Otros parámetros

Se han descrito los principales parámetros que afectan al rendimiento del programa y a la experiencia del usuario que utilice el sistema. No obstante, existen una serie de parámetros que también tienen importancia y que pueden influir en la obtención de una experiencia subjetiva satisfactoria:

1. **Distancia a las pantallas:**
   Esta distancia es directamente proporcional a la relación de aspecto y se ha considerado óptimo el valor para el cual se experimenta mejor sensación de inmersión en la escena sin perder detalle de lo capturado en esta. Este valor ha sido fijado a 3**height* durante la realización de las pruebas anteriores por ser el valor típico para la visualización de contenido 3D, pero el valor más confortable (Tabla 8) dependerá de la relación de aspecto del vídeo en cuestión, del contenido de éste y variará debido a las peculiaridades de los displays inmersivos con respecto a los displays autoestereoscópicos. Las distancias a las pantallas más confortables encontradas para las secuencias de ejemplo utilizadas son:

   Tabla 8 - Distancia más confortable entre la cámara y las pantallas para las secuencias de ejemplo

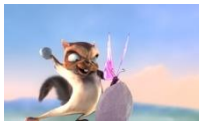
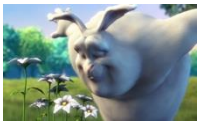
Flowers (arco y lineal)	Butterfly (arco y lineal)	Champagne	Pantomine
*2.5*height*	*2.5*height*	*2*height*	*2*height*

2. **Tipo de array de cámaras:**
   Para las secuencias de vídeo sintético se han realizado las pruebas tanto con un array lineal como con un array en forma de arco (Sección 2.1.3). Los resultados de rendimiento han sido adecuados y prácticamente idénticos en ambos casos. No obstante, al capturar el sistema únicamente el movimiento horizontal de cabeza, resulta más natural el uso de arrays lineales en los que un movimiento de cabeza se traduce en un movimiento del objeto retratado en la escena; en oposición a los arrays en forma de arco en los que un movimiento horizontal de la cabeza se traduce también en una rotación en torno al objeto retratado en la escena.

3. **Rotación:**
   Puesto que se facilita al usuario la posibilidad de activar o desactivar el giróscopo de las Oculus Rift, se ha decidido repetir las pruebas anteriores con el modo de simulación de pantalla desactivado.
   No se han observado cambios perceptibles en el rendimiento ni calidad de imagen, aunque si se ha concluido que la activación del giróscopo ayuda al usuario a situarse en la escena y saber hacia dónde debe mover la cabeza para conmutar entre vistas.

En la Tabla 9 se muestra un resumen de los principales resultados obtenidos tras las pruebas experimentales aquí descritas:

Tabla 9 - Resumen de resultados más adecuados obtenidos en las pruebas experimentales

		BUTTERFLY  Arco / Linear	FLOWERS  Arco / Linear	CHAMPAGNE 	PANTOMINE 
RESOLUCIÓN DE VÍDEO (en píxeles)		640x384	640x384	640x480	640x480
DISTANCIA INTEROCULAR (en nº saltos)	Rango adecuado	5 – 14	3 – 7	1 – 3	1 – 5
	Valor más confortable	9	5	2	2
DISTANCIA DE CONMUTACIÓN (in mm)	Rango adecuado	5 – 15	5 – 15	20 – 50	30 – 60
	Valor más confortable	10	10	30	40
Nº DE CÁMARAS EN REPRODUCCIÓN EN 2º PLANO	Rango adecuado	26 – 46	30 – 50	20 – 40	10 – 20
	Valor más confortable	36	40	30	14
MODO DE REPRODUCCIÓN EN 2º PLANO		Adaptativo	Adaptativo	Adaptativo	Adaptativo
DISTANCIA A LAS PANTALLAS		2.5**height*	2.5**height*	2**height*	2**height*

5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1. Conclusiones

Se ha desarrollado un prototipo de sistema de visualización de secuencias de vídeo multivista para las Oculus Rift, simulando el comportamiento de un display autoestereoscópico. Una pantalla virtual en una escena 3D reproduce un punto de vista mediante un par de vistas estéreo que varían con la posición de la cabeza del usuario.

Este sistema permite variar el punto de vista en función del posicionamiento del usuario y presenta una serie de ventajas frente a los displays autoestereoscópicos para este tipo de aplicaciones:

- Mayor sensación de inmersión
- Ausencia de interferencias entre las vistas izquierda y derecha, o con otras vistas del conjunto
- Posibilidad de ofrecer mayor resolución espacial por vista (frente a displays autoestereoscópicos)
- Facilidad para la presentación de escenas multivistas con un número elevado de vistas (lo cual resulta costoso y tecnológicamente complejo en displays autoestereoscópicos)

Esto crea por tanto un puente que une los HMD, principalmente utilizados hasta el momento para escenas virtuales 360º o juegos inmersivos, con el video multivista.

Se ha desarrollado una interfaz configurable que permite variar los parámetros de visualización para adaptarlos a diferentes secuencias con diferentes configuraciones de cámaras, y conseguir una visualización confortable.

La aplicación desarrollada ha dado buenos resultados durante las pruebas experimentales para las distintas secuencias SMV de MPEG y los rangos de valores adecuados para los distintos parámetros implicados en el sistema pueden verse en la Tabla 9. Trabajando con resoluciones SD, el sistema es capaz de tener en reproducción sincronizada en segundo plano hasta 40 vistas para las secuencias sintéticas. También se ha alcanzado una gran fluidez en la transición entre vistas, con distancias de conmutación entre vistas que alcanzan los 10mm, y que logran que el movimiento de cabeza del usuario y el correspondiente cambio de vista sean coherentes.

Así mismo, al estar basado en una aplicación web, el sistema es fácilmente distribuible, pudiendo ser fácilmente adaptado para ser alojado en un servidor web que permita el acceso remoto. Presenta además la ventaja de ser multiplataforma y de poder ser fácilmente adaptado a otros HMDs.

Finalmente, mediante pequeños desarrollos se podría modelar el movimiento de cabeza del usuario medio que utiliza el sistema, con el fin de empezar a evaluar el comportamiento del usuario con un HMD y optimizar los distintos modos de reproducción disponibles en el sistema.

5.2. Líneas futuras

Respecto a las líneas futuras, al estar los HMD y las aplicaciones de realidad virtual en pleno auge, existen diferentes vías en las que centrar el desarrollo y utilización de este sistema:

1. La utilización de la herramienta desarrollada para la realización de pruebas subjetivas de evaluación de calidad de experiencia para contenidos SMV. Estas pruebas permitirían evaluar la diferencia de calidad de experiencia percibida entre la visualización de contenido SMV en displays autoestereoscópicos y en HMDs.

2. La optimización de los modos de reproducción en segundo plano para que estos se ajusten lo máximo posible a las costumbres de movimiento que tiene el usuario que utiliza el sistema. Para ello, sería interesante poder extraer de la aplicación los datos referentes al posicionamiento de las Oculus Rift en función del tiempo. Con estos datos y tras la realización de pruebas como las descritas en el párrafo anterior, se podría modelar el movimiento del usuario y crear nuevos modos de reproducción en segundo plano más fieles al comportamiento del usuario.

3. La migración de este sistema a otros HMDs. En un primer momento puede parecer lo más razonable el adaptar el sistema a la versión oficial de las Oculus Rift (CV1) que ha empezado a comercializarse a principios de 2016. No obstante, esta vía de desarrollo es irrefrenable, puesto que cada día son más compañías las que se lanzan al desarrollo de un prototipo HMD (HTC Vive, PlayStation VR…). Esto, junto a la integración que presenta Unity con la gran mayoría de HMD, hace que pueda resultar interesante la adaptación del sistema a otros modelos; pudiéndose mantener la gran parte del código y teniendo únicamente que comprobar si las funciones de posicionamiento de usuario son las mismas y, en caso contrario, adaptarlas para el nuevo dispositivo.

4. La adaptación del sistema para mejorar su comportamiento a resoluciones HD.

5. El desarrollo de una aplicación que no necesite ser ejecutada en el navegador. Esto no será posible hasta que el entorno de desarrollo (Unity) permita pasar como parámetro un tiempo a la hora de reproducir una textura de vídeo (Sección 3.2.2.4). No obstante, sería algo interesante a desarrollar una vez sea posible, puesto que una aplicación ejecutable presentaría una serie de ventajas frente a una aplicación web, entre las cuales destacarían la eliminación de:
    - la necesidad de utilizar un navegador y plugins especiales (Sección 7.1).
    - las limitaciones de memoria y rendimiento que presentan la ejecución de este tipo de aplicaciones en un navegador. Esto permitiría aumentar el número de vistas en reproducción en segundo plano (Sección 4.2.4), pudiendo disminuir por tanto la distancia de conmutación entre vistas (Sección 4.2.3). Así mismo, permitiría a su vez la utilización de resoluciones mayores (Sección 4.2.1) que mejoren la calidad de imagen.

6. BIBLIOGRAFÍA

[1] Li Song, Xun Tang, Wei Zhang, Xiaokang Yang and Pingjian Xia, "The SJTU 4K video sequence dataset", 2013 Fifth International Workshop on Quality of Multimedia Experience (QoMEX), p.34-35, Klagenfurt, Austria, 2013.

[2] M. Tanimoto, "Free-viewpoint TV", IEEE 17th International Conference on Image Processing, p.2393-2396, Hong Kong, CH, Sept. 2010.

[3] M. Tanimoto, T. Fujii, T. Senoh, T. Aoki, Y. Sugihara, "Test sequences with different camera arrangements for Call for Proposals on multiview video coding", contrib. M12338, 73th MPEG meeting, Poznan, PL, July. 2005.

[4] N. Dodgson, "Autostereo displays: 3D without glasses", Invited Paper, Electronic Information Displays, Esher, Surrey, Nov. 1997.

[5] Y. Takaki, "Super multi-view display and holographic display", IEEE LEOS Annual Meeting Conference Proceedings, p.12-13, Belek-Antalya, Turkey, 2009.

[6] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, B. MacIntyre, "Recent Advances in Augmented Reality", IEEE Computer Graphics and Applications, Vol.21, Issue.6, p.34-47, Nov. 2001.

[7] Gun Bang, Gauthier Lafruit, Masayuki Tanimoto, "Description of 360 3D video application Exploration Experiments on Divergent multi-view video", contrib. M16129, 114th MPEG meeting, San Diego, US, Feb. 2016.

[8] P. Carballeira, J. Gutiérrez, F. Morán, J. Cabrera, N. García, "Subjective Evaluation of Super Multiview Video in Consumer 3D Displays", International Workshop on Quality of Multimedia Experience, p.1-6, Costa Navarino, GR, May. 2015.

[9] J. Cubelos, P. Carballeira, F. Morán, "SMV player for HMDs", contrib. M38631, 115th MPEG meeting, Geneva, Switzerland, May 2016.

[10] B. Austin Davis, K. Bryla, P. Alexander Benton, "Oculus Rift in Action", Manning, Aug. 2015.

[11] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, C. Zhang, "Multi-view imaging and 3DTV - Special issue overview and introduction", IEEE Signal Processing Magazine, p.10-21, Nov. 2007.

[12] UNITY. Unity3D Documentation. http://docs.unity3d.com

[13] P. Carballeira, "A framework for the analysis and optimization of delay in multiview video coding schemes", Universidad Politécnica de Madrid, 2014.

[14] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, T. Wiegand, "3D video and free viewpoint video - technologies, applications and mpeg standards", IEEE International Conference on Multimedia and Expo, p.2161-2164, Toronto, CA, July. 2006 .

[15] A. Smolic, "3D video and free viewpoint video - From capture to display", El Sevier, Pattern Recognition 44, 15th Sept. 2010.

[16] NOKIA OZO https://ozo.nokia.com/eu/nokia-ozo-specs

[17] LYTRO IMMERGE https://www.lytro.com/immerge#immergerDetails

[18] S. Bing Kang, R. Szeliski, P. Anandan, "The geometry-image representation tradeoff for rendering", IEEE 7th International Conference on Image Processing, p.13-16 vol.2, Vancouver, CA, Sept. 2000.

[19] M.Tanimoto, M. Panahpour Tehrani, T. Fujii, T. Yendo, "A review of the ultimate 3DTV and its related technologies", IEEE Signal Processing Magazine, p.67-76, Jan. 2011.

[20] P. Merkle, A. Smolic, K. Müller, T. Wiegand, "Multi-view video plus depth representation and coding", IEEE 14th International Conference on Image Processing, p.201-204, San Antonio, US, Sept. 2007.

[21] A. Redert, M. Op de Beeck, C. Fehn, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, I. Sexton, P. Surman, "Advanced three-dimensional television system technologies", IEEE First International Symposium on 3D Data Processing Visualization and Transmission, p.313-319, Padova, IT, June. 2002.

[22] G. J. Sullivan, J. Ohm, W. Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Transactions on Circuits and Systems for Video Technology, Vol.22, Issue.12, p.1649-1668, Sept. 2012.

[23] P. Carballeira, J. Gutierrez, F. Moran, J. Cabrera, F. Jaureguizar, N. Garcia, "SMV/FN subjective assessment: MultiView Perceptual Disparity Model", contrib. M38063, 113th MPEG meeting, San Diego, US, Feb. 2016.

[24] N. Holliman, "3D Display Systems", Draft v3, University of Durham, March. 2003.

[25] OCULUS. PC SDK Developer Guide & Best Practices Guide, 0.5. https://developer.oculus.com/documentation/pcsdk/latest/

[26] WAREABLE. Best VR games 2016. http://www.wareable.com/gaming/top-vr-games-for-oculus-rift-project-morpheus-gear-vr-and-project-cardboard

[27] SAMSUNG. Samsung Gear VR. http://www.samsung.com/es/consumer/mobile-devices/wearables/gear/SM-R322NZWAPHE

[28] G. Bang, G. Lafruit and M. Tanimoto, "Description of 360 3D video application Exploration. Experiments on Divergent multi-view video", output doc. N16129, 114th MPEG meeting, San Diego, US, Feb. 2016.

[29] G. Bang, J. Lee, G. Lee, N. Ho Hur, "The head mounted display applications for Free-viewpoint video service", contrib. M36488, 112th MPEG meeting, Warsaw, PL, June. 2015.

[30] OCULUS. Unity Developer Guide https://developer.oculus.com/documentation/game-engines/latest/concepts/book-unity

[31] UNITY ASSET STORE. Simple MovieTextures for Unity WebGL. https://www.assetstore.unity3d.com/en/ - !/content/38369

[32] KHRONOS. OpenGL ES 2.0 for the web. https://www.khronos.org/webgl/

[33] M. Tanimoto, T. Fujii, N. Fukushima, "1D Parallel Test Sequences for MPEG-FTV", contrib. M15378, 84th MPEG meeting, Archamps, FR, April. 2008.

[34] P. Tamas Kovacs, A. Fekete, K. Lackner, V. Kiran Adhikarla, A. Zare, T. Balogh, “Big Buck Bunny lighfield test sequences”, contrib. M36500, 112th MPEG meeting, Warsaw, PL, July. 2015.

[35] MOZILLA NIGHTLY. https://nightly.mozilla.org

[36] MOZILLA. Mozilla WebVR Plus. https://addons.mozilla.org/en-US/android/addon/mozilla-webvr-enabler

[37] GTK2K. Unity WebVR Test Assets. https://github.com/gtk2k/Unity-WebVR-Test-Assets

[38] XIPH. The Ogg cointainer format. https://xiph.org/ogg

[39] FFMPEG. LibTheora. https://ffmpeg.org/ffmpeg-codecs.html#libtheora

7. APÉNDICES

En esta sección se van a describir los distintos requisitos que presenta el sistema, tanto el navegador y el plugin necesarios para su correcta ejecución, como el formato de archivo necesario para los vídeos de las secuencias de entrada. Finalmente, se presentará un completo manual de usuario en inglés e información sobre el contenido incluido en el CD adjunto a este documento.

7.1. Requisitos del navegador

Como ya explicamos en la sección de sincronización de vistas (Sección 3.2.2.3), durante el desarrollo del sistema aquí descrito, se ha utilizado un paquete que implementa soporte para texturización de vídeos en Unity WebGL. Aunque lo que realmente nos interese de este paquete sea la función de texturización que permite reproducir un vídeo en un momento concreto, facilitando así la tarea de sincronización entre vistas, la utilización de este paquete exige que el proyecto final sea exportado para WebGL.

La realidad virtual se encuentra en pleno auge y todavía son pocos navegadores los que dan soporte a contenido WebGL y a su vez permiten la integración con un HMD. A día de hoy, los dos principales proyectos que cumplen ambos requisitos son:

- Google Chrome con el proyecto Chromium
- Mozilla con la versión experimental del navegador Mozilla Nightly [35]

Durante el desarrollo del sistema se ha decidido utilizar Mozilla Nightly, versión experimental del conocido navegador Mozilla Firefox que se encuentra en fase de prueba y que recibe actualizaciones diarias.

Adicionalmente a la instalación de Mozilla Nightly, es necesaria la instalación de un plugin que habilite la integración con un HMD, llamado “Mozilla WebVR Enabler” [36], instalado por defecto en las últimas versiones del navegador. Este plugin da soporte a los dispositivos de realidad virtual como las Oculus Rift para aplicaciones web, permitiendo a los desarrolladores crear contenido de realidad virtual que corra en cualquier navegador que permita WebGL.

Durante la realización de este proyecto se ha utilizado la versión 45.0.a1 del navegador Firefox Nightly y, las versiones 0.2.0 y 0.5.0 del plugin WebVR. No obstante, el sistema debería ser ejecutable y compatible para versiones posteriores de las aquí mencionadas.

A día de hoy, Unity no da soporte a las Oculus en su formato de salida para WebGL. Por tanto, se ha necesitado un paquete extra que integra las funciones de posicionamiento de un HMD de Unity con las funciones características del plugin WebVR [37].

7.2. Preparación de los archivos de entrada

Antes de ejecutar el programa, hay que asegurarse de tener los vídeos en el formato adecuado para el correcto funcionamiento del sistema. El entorno Unity trabaja con el formato Ogg Theora

video (también llamado OGV) [38], códec de vídeo libre con bajo consumo de CPU, que ha sido el utilizado por tanto durante todo el desarrollo.

Por tanto, los vídeos deberán de ser convertidos al formato OGV antes de ejecutar el programa; pudiéndose aprovechar esta fase para la conversión a resoluciones SD.

Para la conversión se ha realizado usando el *preset* de máxima calidad de vídeo con *bitrate* variable de la librería *theora*. No obstante, y tras comprobar que Unity es capaz de trabajar con archivos con otro tipo de codificaciones únicamente encapsulados con OGV, en un futuro podría adaptarse el sistema [39].

7.3. Manual de usuario

USER MANUAL

SUPER MULTIVIEW PLAYER FOR HMDs

E.T.S.I Telecomunicación – Universidad Politécnica de Madrid

Javier Cubelos Ordás

This document is a complete user manual of a prototype of a HMD-based visualization system for SMV content. The system simulates the visualization of an autostereoscopic display, in a 3D scene in which the user is immersed. The system takes a set of multiview sequences as input and, depending of the user's head position, it will vary the user's viewpoint adapting it to this position

1. REQUIREMENTS

a. Computer

This system has been tested with a computer with the following characteristics:

- Intel® Core™ i7-4790 CPU @ 3.60 GHz
- RAM 16GB
- 64-bit OS
- NVIDIA GeForce GTX 970 (4095 MB) updated to 361.91

The use of a computer with worst or different characteristics can be translated into a performance deterioration. It has been proved that the system doesn't work properly for older versions than the 358.70 one of the NVIDIA driver.

b. Oculus

The computer used for executing the system has to have installed the Oculus runtime. The system has been developed using the 0.8 version and it has been proven that it doesn't work properly for older versions. The runtime has to be initialized before executing the system.

c. Browser

For the correct execution of the system, it's important to use a browser that supports WebGL and integration with HMDs. We recommend the use of Firefox Nighly in its 45.0.a1 version, and install the "Mozilla WebVR Enabler" addon (we recommend the use of the 0.5.0 version). The system should work for newer versions.

2. VIDEO FILE CONVERSION

The input videos of the multiview sequence have to be in the OGV (Ogg Theora Video) format. A way to convert them from a common format as AVI is using the ffmpeg2theora converter, for example if the input files are in AVI format:

for %%A IN (.avi) DO ffmpeg2theora.exe -v 10 "%%A"*

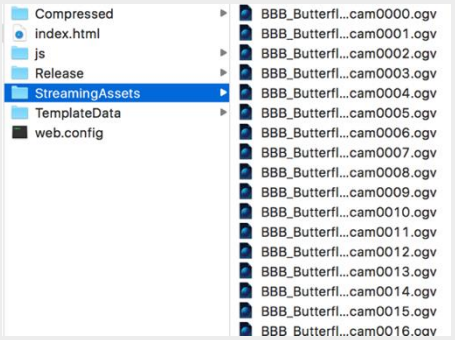
If the videos are in YUV instead of AVI, we recommend you to use the ffmpeg converter before doing the previous step, for example:

for %%A IN (.yuv) DO ffmpeg.exe -s 1280x768 -r 25 -i "%%A" -pix_fmt yuv420p -vcodec rawvideo -r 25 -s 1280x768 "%%ñA.avi"*

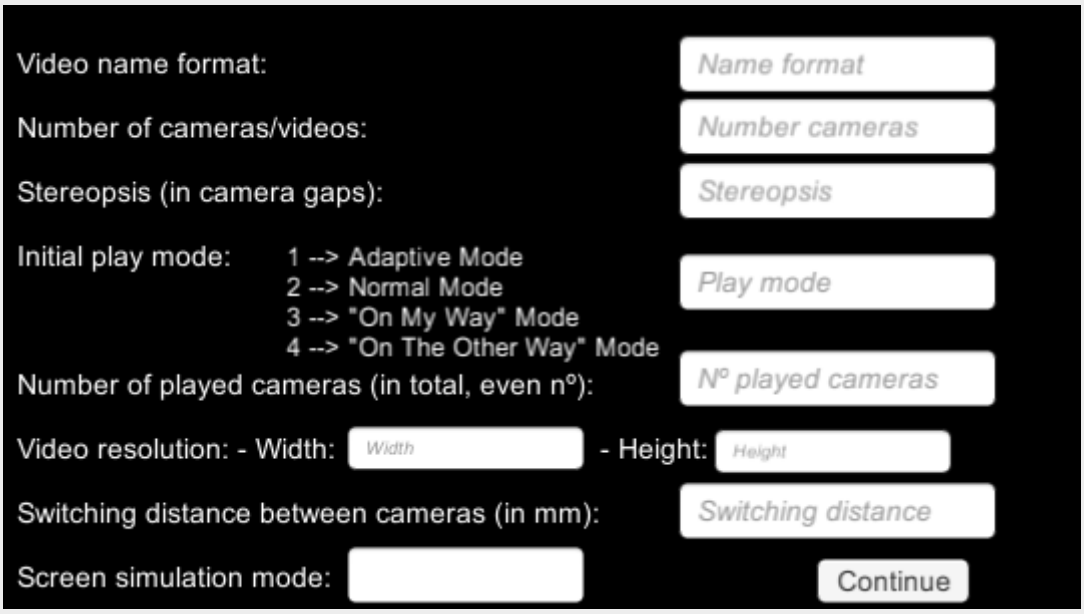
(Check the details of these converters in their websites)

3. VIDEO FILE INSERTION

Before executing the program, is important to put all the videos in the folder 'StreamingAssets'. All the video files must follow the same name format and be numbered at the end of the name with 4 digit, starting from 0000.



4. SYSTEM CONFIGURATION



After including the videos in the 'StreamingAssets' folder and opening the 'index.html' file with the correct browser (see section 1.c), it's time to configure the system:

- Video name format: Prefix of the video files names common in all the files. For example, for the sequence of 91 videos with videos from "BBB_Butterfly_cam0000" to "BBB_Butterfly_cam0090", the video name format will be "BBB_Butterfly_cam"
- Number of cameras/videos: The total number of videos in the sequence

- Interocular distance (stereo baseline): distance between left and right views (measured in number of camera gaps).

- Number of background views: number of views that are played simultaneously in the background, and prepared to be displayed on the screen. The value of this parameter does not include the number of views covered by the interocular distance that will be also played in the background

- Different prediction modes for the "background" views (*Initial play mode)*: The system has different ways of preparing the videos next to the user's position for improving his experience. Although we recommend the use of the 'Adaptive Mode', it can be sometimes useful to use one of the other modes:
    - *Manual modes:*
        - *Normal mode*: the same number of views set in both directions (taking the user current position as center).
        - *On my way / On the other way modes*: decide the quantity of views to prepare depending of the user's movement direction.
    - *Adaptive Mode:* it sets the same number of views in both directions (like the *normal* mode) but adapts this ratio at the limits of the camera rig setting more views in the opposite direction. The *adaptive* mode is most useful in almost all the situations.

- Video resolution: Resolution of the input videos. We recommend the use of SD resolutions.

- View switching distance: horizontal distance between two consecutives views, i.e. distance that the head needs to move to switch to the contiguous view.
- Rotation (*Screen simulation mode*): the user has the possibility of turning **off** the rotation tracking of the Oculus Rift in order to make the screens "follow" the head orientation, or keep it **on** fixing its position in the 3D scene (screen simulation mode).
- Distance to screen: measured relatively to screen height (vertical resolution of the sequence). This parameter is only adjustable during execution time.

We recommend to try a few times changing the values of the 'number of played cameras' and the 'switching distance between cameras' because the optimal values of these parameters can change a lot depending of the computer and the sequence of videos used.

5. SYSTEM EXECUTION

Once the system has been configured and after clicking the 'continue' button, the system will be executed. Now it's time to switch on the Oculus Rift:

1. Click on the 'Oculus' logo in order to switch to Oculus Mode. In this mode the oculus tracker is activated and the result can be observed in the screen of the computer but not in the Oculus Rift
2. Click on the 'Fullscreen' logo in order to activate the Oculus Display. Now it's time to wear the Oculus.

Sometimes it can be usefull to reset the Oculus sensor after these steps clicking the 'Reset Sensor' button.

Now the system should be completely operable. We recommend to move the head slowly (to the left and right) to feel the best experience possible. If you see desynchronization between both views maybe you should change the parameters (restarting the system) or the play mode (automatic mode recommended).

During the execution you can change some settings using the keyboard:

- Change the 'play mode':
  - '**F1**' key: switch to 'Adaptive mode'
  - '**F2**' key: switch to 'Normal mode'
  - '**F3**' key: switch to 'On My Way mode'
  - '**F4**' key: switch to 'On The Other Way mode'

- Move the screens:

  - '**Up**' key: zoom out
  - '**Down**' key: zoom in
- Modify the user position with the keyboard (if the Oculus doesn't work correctly):
  - '**A**' key: move the user to the left
  - '**D**' key: move the user to the right
- '**S**' key: force synchronization
- '**TAB**' key: switch on/off the console

7.4. Contenido del CD adjunto

Junto al presente documento, se adjunta un CD con el siguiente contenido:

- Fichero **"User Manual.pdf"** que contiene el manual de usuario en inglés descrito en el apartado anterior, el cual explica todos los pasos necesarios para la correcta ejecución del sistema.
- Carpeta **"Requirements"** que contiene las siguientes sub-carpetas:
    - Carpeta **"Oculus Runtime"** que contiene el fichero de instalación del Oculus Runtime para Windows utilizado durante la realización de este proyecto (versión 0.8).
    - Carpeta **"Firefox Nightly"** que contiene el fichero de instalación del navegador Firefox Nightly para Windows utilizado durante la realización de este proyecto (versión 45.0.a1) e instrucciones sobre la instalación del plugin necesario.
    - Carpeta **"FFmpeg"** que contiene los ejecutables de los dos conversores utilizados en este proyecto (ffmpeg y ffmpeg2theora) junto a un ejemplo de archivo BAT para la conversión de YUV a OGV con estos dos conversores.
- Carpeta **"Unity Project"** que contiene el proyecto completo de Unity para futuras modificaciones del sistema (los scripts con el código comentado se encuentran en la subcarpeta *Assets*)
- Carpeta **"Export"** que contiene el sistema final implementado, que contiene una serie de archivos y carpetas entre las que cabe destacar:
    - Carpeta **"StreamingAssets"** en la que se incluyen las secuencias SMV de MPEG utilizadas de ejemplo, y en donde, si se desease utilizar otras secuencias, se deberían insertar los vídeos de la secuencia ya convertidos a formato OGV y siguiendo la numeración escogida. Adicionalmente, se ha decidido dejar como ejemplo los vídeos de la secuencia *BBB_Butterfly_cam*, los cuales pueden ser eliminados y cambiados por nuevas secuencias.
    - Archivo **"index.html"** que ejecuta el sistema en el navegador (abrir en Firefox Nightly).